Classifying Cyber Aggression

in Social Media Posts

A thesis

submitted in partial fulfilment

of the requirements for the Degree of

Master of Applied Science

at

Lincoln University

by

Meisy Fortunatus

Lincoln University

2019

Abstract of a thesis submitted in partial fulfilment of the
requirements for the Degree of Master of Applied Science.

Identifying Elements of Cyber Aggression in Social Media Posts

by

Meisy Fortunatus

Cyber aggression is one of the most prevalent issues stemmed from the growing number of internet users. Given the numerous amount of online posts every day, it is not feasible to detect textual cyber aggression manually. This research focuses on analysing social media posts to find elements of cyber aggression and then build an algorithm that uses these elements in a set of rules to detect cyberbullying effectively. Lexicon enhanced rule-based method is used to detect cyber aggression on three different types of social media textual communication: single post from Facebook, question and answer pair from Formspring.me, and thread style from MySpace.com. The algorithm is evaluated using a combination of accuracy, precision, recall, and F1 measure. It was found that the algorithm performed best for single style data and the least for thread style data.

Keywords: cyberbullying, cyber aggression, textual aggression, harrassment, sentiment analysis, emoticon sentiment, emoji sentiment, lexicon, social media, aggression detection

Acknowledgements

First and foremost I would like to give thanks to Lord Jesus Christ for His guidance throughout this project. I would also like to thank my supervisors Patricia Anthony and Stuart Charters for their helpful feedback during the duration of my thesis. Next, I would like to express my gratitude to MFAT and Lincoln University for giving me this opportunity. For all the important people in my life who have been giving me supports all this time: my family and friends, and some people who fit as both.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The technology is striving to improve human life, with a common goal to make everything easier and faster than ever. The Internet has proven to be one of the most prominent technology, improving many aspects including information gathering and communication. Nowadays, the internet has become an integral part of human life and having access to it has become natural (Talwar, Gomez-Garibello, & Shariff, 2014). Combined with the rapid development of portable communication devices and mobile networks, people now have internet access anywhere anytime.

This ease of access contributes to the rapid growth of internet users over the years and along with it, challenges arise. One of these challenges is cyber aggression, which can be defined as a voluntary act of hurting other people using technological communication media (Hinduja & Patchin, 2008; Schoffstall & Cohen, 2011; Whittaker & Kowalski, 2014). Communication methods vary from a (cellular) phone, online game, social media, email, and many more (Hinduja & Patchin, 2008). Borrowing from the definition of traditional bullying, cyberbullying can be defined as repetitive acts of cyber aggression, because repetition is a benchmark that discerns bullying from aggression (Pettalia, Levin, & Dickinson, 2013; Slonje, Smith, & Frisén, 2013). However, the terms cyberbullying and cyber aggression have been used interchangeably in past studies due to the lack of a formal definition of cyberbullying.

Cyberbullying cases are prevalent and it has encouraged research towards a better understanding of its nature. Whittaker and Kowalski (2014) reported that out of 244 respondents, 18.2% admitted to being a victim of cyber aggression at least once a year. Other research indicated that there was 24% victimization of cyberbullying on average (Patchin & Hinduja, as cited in Bastiaensens et al., 2014).

Victims of cyber aggression suffer from various impacts, from school issues (Hinduja & Patchin, 2008), depression (Bauman, Toomey, & Walker, 2013), substance abuse (Hinduja & Patchin, 2008), and suicide attempts (Bauman et al., 2013). While many might argue otherwise, research has also found that 35.5% of respondents perceived that the effect of cyberbullying is just as severe as traditional bullying (Pettalia et al., 2013).

Despite the negative impact of cyber aggression, victims are usually reluctant to report the incidents (Dehue, Bolman, & Völlink, 2008; Slonje & Smith, 2008) due to various reasons; fear of having their electronic devices taken away (Parris, Varjas, Meyers, & Cutts, 2011; Stacey, 2009), anxious that others might not take the issue seriously (Parris et al., 2011), fear that nobody can stop it from

reoccurring (Smith et al., 2008), even sceptical that the perpetrator will be punished (Pettalia et al., 2013).

1.1 Cyber aggression in online social media

Online social media has become an important platform for communication. People share information, establish connections, and even form relationships via social media. Classifying social media is not an easy task, given the numerous different social media platforms, each of which has its own functionality and capacity (Hanna, Rohm, & Crittenden, 2011; Kietzmann, Hermkens, McCarthy, & Silvestre, 2011). Kietzmann et al. (2011) divided them into several categories: general masses platform (Facebook and Friendster), professional networking platforms (LinkedIn), media sharing platforms (YouTube, Flickr, MySpace), weblogs, social news and bookmarking (Reddit, Digg), and microblogging (Twitter). Social media platforms offer many types of communication as part of their features: 1 on 1 or group chat; question and answer pair; thread, post and comments; even comments and its child comments.

A web information company, Alexa, released a list of top 10 global websites by late 2010, which included Facebook, YouTube, Blogger.com, Windows Live, Twitter, and QQ.com, which are social media sites (Hanna et al., 2011). Twitter and Facebook could be seen as examples of the popularity of social media, where it was reported that on average there were 500 million tweets each day by 2018 (Aslam, 2018) and 1.4 billion active users by December 2017 on Facebook ("Facebook company info," 2018).

Although different social media platforms have their own scope, there are uniform features that each of them offers: anonymity and freedom of speech. Anonymity in which users are free to enter their personal data without official clarification whatsoever and they are given freedom to post anything. Unfortunately, these features have been used by irresponsible parties to demonstrate unacceptable social behaviours (Ruths & Pfeffer, 2014; Yin, Xue, Hong, Davison, & Edwards, 2009). Now that cyber communication is a common occurrence, these unacceptable social behaviours can be seen as an extension of rudeness in offline intrapersonal relationships (Yin et al., 2009).

A study showed that the media through which cyber aggression happens the most corresponds to the most used social media (Whittaker & Kowalski, 2014). Social media platforms go in and out of fashion, thus researching human behaviour through social media cannot rely on a single data set only (Ruths & Pfeffer, 2014). Unfortunately, past research on cyberbullying detection was usually focused on a dataset taken from a single platform, thus it would not be applicable to other platforms.

1.2 Emoticon and emoji in online social media

In traditional face to face communication, people use non-verbal cues like gestures and facial expressions to accompany words. The lack of these non-verbal cues is the reason why textual cyber communication in social media can be misleading. To replace non-verbal cues in online communication, people started to use emoticons and more recently, emoji (Oleszkiewicz et al., 2017).

An emoticon can be defined as a set of characters used to depict a facial expression such as ":)", which represents a smiling face (Hogenboom et al., 2013; Kralj Novak, Smailović, Sluban, & Mozetič, 2015; Oleszkiewicz et al., 2017). People can use emoticons to either disambiguate, express, or intensify the feelings or mood that they intend to convey online (Hogenboom et al., 2013; Marengo, Giannotta, & Settanni, 2017; Oleszkiewicz et al., 2017).

Just as technology develops over time, people eventually started to improve emoticons and came up with emoji, which can be described as an advanced graphical representation of human expressions and other objects or concepts (Kralj Novak et al., 2015). The same research explained how emoji complements textual online communication by providing a wide range of expressions, food, animals, places, and many other objects. Initially, when people want to incorporate emoticons in their text, they need to know the characters' combination and type them in manually. Fortunately, the developers have made it easier by providing a list of usable emojis, which users can easily pick and click. Automatic conversion from popular emoticons to their respective emojis is also very common nowadays. The latest version of emoji (version 11.0), lists a total of 2,789 emojis ("Full list of emoji v11.0," 2018).

Emoticons and emoji have become prevalent in online communication such as emails, social media, and text messages (Kralj Novak et al., 2015). Facebook even released a newsletter reporting that more than 60 million emojis are used every day on Facebook (Cohen, 2017). This popularity has paved a new means to dig information online, from personality assessment (Marengo et al., 2017), sentiment analysis (Hogenboom et al., 2013; Kralj Novak et al., 2015), and text polarity (Hogenboom et al., 2015). In accordance to these applications, researchers have been conducting research towards a better understanding of emoji, including what is claimed as the first emoji sentiment lexicon, the Emoji Sentiment Ranking (Kralj Novak et al., 2015).

1.3 Importance of automated cyber aggression detection

As discussed, the popularity of online social media has resulted in large numbers of posts made every day making it almost impossible to manually detect cyber aggression. This is why automated cyber aggression detection is needed. While emoticons have been incorporated in a few studies of aggression detection, emojis are not very well explored (Bayzick, Kontostathis, & Edwards, 2018; Hogenboom et al., 2013; Hogenboom et al., 2015; Kralj Novak et al., 2015; Nalinipriya & Asswini, 2015; Nandhini & Sheeba, 2015; Yin et al., 2009).

Cyber aggression detection can be seen as an extension of sentiment analysis, which is a task that classifies an input as either positive, neutral, or negative. Negatively labelled inputs can be divided further into four categories of basic human emotions (Ekman & Friesen, 1971): sadness, anger, disgust, and fear. In general, not all negative emotions can be perceived aggressive, such as sadness and fear. Thus aggression detection extends sentiment analysis by identifying aggression elements from an input.

This research will develop an algorithm to analyse social media posts with text, emoji, and emoticons to find elements of cyber aggression. Correctly identifying elements of aggression will, in turn, provide a better means to detect, prevent, and stop cyberbullying.

1.4 Research Questions and Objectives

1.4.1 Research Questions

In order to achieve the objectives of the research, some research questions have been derived:

1. How do we identify elements of cyber aggression in three different types social media text (single post, question & answer, and thread)?
    i. Which social media should we choose as dataset source?
    ii. How do we apply combination text, emoji, and emoticon sentiment in an algorithm to identify elements of cyber aggression?
2. How do we validate the performance of the algorithm resulting from (1)?

1.4.2 Research Objectives

The objectives of this research are to:

1. Identify existing suitable techniques that can be used to detect cyber aggression in a social media post (assuming that social media post contains a combination of text, emojis, and emoticons).
2. Develop an algorithm to detect cyber aggression in a social media post.
3. Measure the performance of the proposed algorithm using different types of social media text: single post, question & answer, and thread.

1.5 Thesis Outline

The remainder of the thesis is structured as follows. Literature review with thorough discussion of related past studies is in Chapter 2. Chapter 3 introduces this research's questions and objectives. In-depth discussion on the methodology used in this study can be found in Chapter 4. The next chapter presents the result of the study, gives a comparison of performances of algorithm throughout the development process, and discuss the given results. Lastly, the conclusions of the study are discussed along with future study prospect.

Chapter 2

Literature Review

Evaluating past research on cyber aggression detection through sentiment analysis and other related research will give a better understanding of the best approach for this study. This chapter is divided into five sections; first, performance measures that are used during the research process is discussed. Second, discussion of some well-known classification methods that have been used in past research on text sentiment analysis and aggression detection. Emoji and emoticon sentiment analysis are discussed in the third section and challenges in analysing text from social media are in the fourth section. Lastly, inter-annotator agreement is reviewed.

2.1 Accuracy, Precision, Recall, F Measure

In data science, accuracy, precision, recall, and F measure are among the most used metrics to indicate the performance of a system. Performance of binary classification task such as aggression detection can be measured using these four metrics, allowing researchers to compare which model works best (Koo, 2018). This research utilizes these measurements to assess the performance of different algorithms used to identify cyberbullying text. Calculating accuracy, precision and recall is simple, by utilizing the true/false positive/negative count. Table 2.1 shows the definition of true/false positive/negative.

Table 2.1 Relationship of actual and prediction result

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

True positives can be defined as entries that are correctly identified as bullying while false negatives refer to entries that are incorrectly identified as bullying. On the other hand, false positives are innocent entries incorrectly identified as bullying and true negatives are entries that are correctly identified as innocent.

2.1.1 Accuracy

Accuracy is the most straightforward approach, being the ratio of correctly identified entries to the total number of data entries. It is calculated by adding the number of true positive and true negative divided by the total of all entries.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

Although it seems to give an accurate performance measure, when a dataset is not symmetrical (having a similar amount of positive and negative data), accuracy cannot be used as a sole measurement of performance. This is due to the nature of accuracy that puts a high emphasis on the sum of correctly identified entries that software will get a high accuracy score even though it is falsely labelling entries as aggression. For example, a dataset consists of 96 aggressions and 4 innocent entries and the system can correctly identify all 96 aggressions but missed all innocents, the accuracy will still be 96% even at the cost of being unable to identify the 4 innocent inputs (Koo, 2018).

2.1.2 Precision

Precision shows the ratio of inputs correctly identified as positive to the sum of inputs labelled as positive. This calculation gives an insight on how well the system can correctly identify positive inputs. Precision is widely used as a good performance measurement towards a system that is emphasized on avoiding false positive (Koo, 2018).

$$Precision = \frac{TP}{TP + FP}$$

Precision is calculated as the total of true positives divided by the sum of true positives and false positives.

2.1.3 Recall

Recall or sensitivity is the ratio of inputs correctly identified as positive to the sum of actual positive inputs. Recall gives information on how many of all positive inputs can the system correctly identify. As oppose to precision, recall is usually used to measure a system where avoiding false negative is the main concern (Koo, 2018).

$$Recall = \frac{TP}{TP + FN}$$

The recall score is calculated as the number of true positives divided by the sum of true positives and false negatives.

2.1.4 F1 Measure

The F1 measure (also known as F1 Score, F Measure, or F Score) is the weighted average of precision and recall (Joshi, 2016), giving a more balanced approach towards recall and precision (Koo, 2018). Thus the F measure is highly regarded as a measure when working with an asymmetrical dataset.

$$F1\ Score = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

F1 measure is calculated by multiplying precision and recall by two and then divide the result with the sum of precision and recall score.

2.2 Classification methods and algorithms

Detecting cyber aggression can be considered as an extension of sentiment analysis since both tasks deal with human emotions detection (Gordeev, 2016). Sentiment analysis can be defined as a task to classify a text to its sentiment, which can be either positive, neutral, or negative (Devika, Sunitha, & Ganesh, 2016).

Sentiment analysis is mainly divided into three approaches namely rule-based, machine learning, and lexicon-based (Devika et al., 2016). These approaches are discussed in sections below.

2.2.1 Rule-based Method

Rule-based method for sentiment analysis is probably the most straightforward and easy to comprehend method. It is indicated by enforcing a set of rules to determine the sentiment of a given input (Devika et al., 2016). Furthermore, Devika et al. (2016) described that the set of rules are used to calculate the sentiment of the text, usually ranging from a negative to a positive number, to illustrate the polarity and enmity of the input. Since the rules can be tailored to detect certain elements, this can be considered as the easiest method to detect cyber aggression.

Bayzick et al. (2018) used the rule-based method on research to detect cyber aggression, with dataset taken from MySpace.com. The result of their research was a program called BullyTracer, designed to detect the presence of cyberbullying in a conversation. The rules set in the program was quite straightforward, where every post containing bad words like swear words and insults, using a lot of capital letters, and addressed to a second person pronoun were labelled as bullying. After

evaluation, they found out that the rules were enough to capture the elements of aggression but too broad that it also marked non-aggressive inputs as bullying.

Another study by Nalinipriya and Asswini (2015) was focused on the detection and prevention of cyberbullying occurrence by establishing a set of rules against “blocked words”, consisting of swear words from noswearing.com. The researchers proposed this approach to prevent cyberbullying by blocking certain words and sending a warning to users whenever a text containing blocked words is to be posted. As a follow-up strategy, the proposed system would send an alert to trusted contacts of the receiver. The result of the research was a dummy social media embedded with the proposed cyberbullying prevention system. It was unfortunate that the module cannot be implemented without involving social media providers, who might be reluctant to apply the module because it might impair the appeals of anonymity and freedom that social media promotes.

On another study of harassment detection, three different features were utilized: local, sentiment, and contextual features (Yin et al., 2009). Yin et al. described local features as features extracted from the input itself, assessed using TF-IDF (term frequency–inverse document frequency) rule. For sentiment features, the researchers established rules to detect patterns of harassment texts, which highlighted the connection between bad words and personal pronoun. Contextual features were features used to discern the context of the input, to anticipate non-aggressive inputs which can be misinterpreted as harassment, like jokes between friends and a heated argument. Combining these three features, the research found that the program performed better compared to elementary TF-IDF.

Reynolds, Kontostathis, and Edwards (2011) compared rule-based learning method with bag-of-words model using data taken from Formspring.me, a question and answer based social media. The rules applied consisted of “bad words” identification and anonymity check because they argued that anonymity might promote the tendency to perform aggression. They also argued that the most important measure of cyberbullying detection would be the recall or how well could the system identify bullying, meaning precision (how well could the system distinguish non-cyberbullying) was not the main concern. This research claimed to achieve a recall of 96.6%.

Reflecting on past research discussed above, it can be concluded that presence of bad words and exaggerated usage of capital letters have been considered as signs of aggression because bad words like swear words and insults have been used with ill intent and if a text has had more than 50% capital letters, it might be considered offensive and rude (Bayzick et al., 2018; Nalinipriya & Asswini, 2015).

The rule-based approach is known to be robust but highly dependent on the defining rules (Devika et al., 2016). When an established set of rules is too simple, it may result in a large number of false positive results, hence a refined approach is needed to compile the rules. The rules are usually hardcoded to the algorithm which simply means that human intervention is highly needed to recompile new rules.

2.2.2 Machine Learning Method

Machine learning approach essentially handles text processing by getting trained beforehand using a certain learning algorithm (Devika et al., 2016). It can be classified as either supervised or unsupervised machine learning. A set of labelled training data, which contains a pair of input and its expected output, is needed in supervised learning. On the other hand, when a set of input without its expected output is being used to train the machine, it is called unsupervised learning.

Generally, the machine learning algorithm is trained with a set of training dataset and after it has reached a certain state it will then be used to process real data. Due to this nature, when a machine was trained in a certain domain, it would not perform as effectively when processing input from other domains (Devika et al., 2016; Saif, He, Fernandez, & Alani, 2016).

Further review regarding the classifiers of the machine learning approach is as follows:

Neural Network

The neural network is a machine learning technique inspired by how human brain works, by mimicking the way brain neuron works, making impressive and connected indices and it is applicable to both vector presentation and sentence classification (Tul Ain et al., 2017). In sentiment analysis task a neural network system will assume a set of rules based on the training data set, and the rules will then be used to classify an unlabelled input (Nielsen, 2018; Tul Ain et al., 2017).

Over the years, the neural network has opened up a new way to approach Natural Language Processing, and one of the most famous and referred to is Word2Vec. Word2Vec is a project that focused on producing a continuous vector from a word. It has changed the way people see a word vector by introducing a simple method to analyse the semantic and syntactic relationship between words (Mikolov, Chen, Corrado, & Dean, 2013). This method claimed that a simple learning method can also produce high-quality word vector (Mikolov, Chen, et al., 2013).

The vector produced by Word2Vec provides information on the semantic relationship so it can assist with Natural Language Processing and it has been used to assist a lot of other research ever since (Mikolov, Chen, et al., 2013). Though some people argued that another approach provides better accuracy, it was also stated that cost-wise, Word2Vec was still very much winning (Konopík & Pražák, 2015).

Word2Vec was so simplified that it was able to process an accurate high dimensional word vector from a large dataset, even when it was extended to process phrases (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Even with the extension, the model architecture was still so computationally efficient that it can be trained on bigger data compared to previous research (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013).

As discussed before, Word2Vec has become an important research that provides a new way to assess the similarity between words, thus contributing to the success of classification tasks using other classifiers.

Naïve Bayes

Naïve Bayes is a supervised, probabilistic classifier based on Bayes' theorem (Devika et al., 2016; Singh, Singh, & Singh, 2017). In order to classify the input's sentiment, it uses a conditional probability determined by Bayes' rule below

$$P(Sentiment/Sentence) = \frac{P(Sentiment)P(Sentence/Sentiment)}{P(Sentence)}$$

Where P(Sentence/Sentiment) is the product of P(Token/Sentiment) or sentiment derived from the tokens of the sentence (Devika et al., 2016).

In research regarding machine learning classifiers, Singh et al. (2017) showed that Naïve Bayes was arguably fast, though the accuracy was not the highest out of all other classifiers tested. It is popular for sentiment analysis because it is considered to be simpler than other machine learning classifiers and is intuitive (Devika et al., 2016), and only requires a small training dataset (Singh et al., 2017).

Despite its strengths, Naïve Bayes has a limitation which lies in its assumption of independent predictors among the linguistic features (Devika et al., 2016; Ray, 2017).

Support Vector Machine (SVM)

SVM is a supervised non-probabilistic machine learning classifier (Devika et al., 2016) that is famous for sentiment analysis. Hsu, Chang, and Lin (2003) proposed the proceeding for beginners in an SVM practical guide as:

1. Vectorize the input, because SVM demands that input data is represented as a vector of real numbers. To get the vector representation, each word in a text input is converted into a number by following a certain categorization rule. For example, if there are three categories of (apple, orange, banana) then the representation would be (1, 0, 0), (0, 1, 0), and (0, 0, 1).

2. Apply a simple scaling, linear scaling recommended, to avoid domination of bigger numerical attributes over smaller ones and to simplify the calculation.

3. Choose an appropriate model. RBF kernel is suggested for beginners because it has several advantages such as the ability to handle nonlinear cases, has fewer hyperparameters so it is less complex, and has less numerical difficulties.

4. Conduct a cross-validation to determine the best value for parameters needed for RBF kernel: C and γ.

5. Use the result from (d) for the training process.

6. Conduct testing.

It was also pointed out that the procedure might only work best when the given features are limited in number, hence when handling data with a large number of features, a feature handling process is needed before classification with SVM (Hsu et al., 2003).

That is why SVM is mainly used alongside with feature extraction and selection techniques. In a research towards aspect based sentiment analysis, SVM was one of the base learning algorithms with CRF (Conditional Random Field) and ME (Maximum Entropy), where after a feature selection process, the extracted features were processed using an ensemble of all three base algorithms (Akhtar, Gupta, Ekbal, & Bhattacharyya, 2017). Another research by Albertini, Zamberletti, and Gallo (2014) on short document sentiment classification, the output from unsupervised feature extraction using Growing Hierarchical Self-Organizing Map was used as the input for SVM classifier, where it was claimed to perform excellently.

Random Forest

Random Forest classifier is a learning method that works using a decision tree built based on the benchmark that is specified during training stage (Gordeev, 2016). Gordeev conducted a research on aggression detection in English and Russian using Word2Vec to find similarities between words and Random Forest as a classifier. It was found that while processing English inputs yield a decent result of 88%, the performance of Russian language processing was not satisfactory. The possible cause of this gap in performance is because the Russian language has more complex grammar and syntax compared to English (Gordeev, 2016).

Comparing Random Forest to deep learning classifiers also showed that deep learning classifiers are more satisfying (Gordeev & Potapova, 2016) and Gordeev (2016) actually suggested to switch the classifiers to SVM or other neural network-based methods.

Deep Learning

Deep learning is a subset of machine learning, which, according to Day & Lee, is also known as Deep Neural Network (Tul Ain et al., 2017). Ever since it was proposed in 2006 by G.E. Hinton, deep learning has been utilized for speech recognition, natural language processing, and computer vision (Tul Ain et al., 2017). Deep learning differs from the normal neural network by having many layers of non-linear units, enabling it to learn complicated decision sets (Shirani-mehr, 2015).

On research of sentiment analysis of movie reviews using deep learning, Shirani-mehr (2015) compared three different methods of deep learning namely recursive neural network, recurrent neural network, and convolutional neural network (CNN) with Naïve Bayes as a baseline. Shirani-mehr concluded that recurrent neural network gave a similar performance compared to the chosen baseline method because of its inefficiency in representing structural and contextual properties of input sentences. Recursive neural network, which is built like a parse tree, was more able to work on words' relations, thus resulting in better performance compared to recurrent neural network. CNN, which is generally a generalized version of recursive neural network, has the same weakness with recurrent neural network which is missing out on phrases connection. However, this weakness was overcome by using Word2Vec's word vectors, resulting in significant improvement of performance (Shirani-mehr, 2015).

Gordeev and Potapova (2016) researched aggression detection and movie reviews sentiment using CNN, and it was compared to Random Forest classifier. The research used 4chan.org and 2ch.hk as corpora, both were considered as very aggressive communities. While CNN worked best for movie

reviews' sentiment analysis and Russian language aggression detection, Random Forest performed better for aggression detection in English (Gordeev & Potapova).

In general, machine learning needs a lot of training data in order to be able to do its job with good performance and when it is used for scanning inputs outside its trained field, the accuracy will drop significantly (Devika et al., 2016). Moreover, this method does not give sufficient information as to how a result is calculated because it basically works by abstracting the sentence, disregarding the sentence structure (Hogenboom et al., 2015).

2.2.3 Lexicon-based method

Lexicon-based or dictionary-based methods work by using a sentiment lexicon to weigh each word (and phrase) and aggregate the scores of individual words, where both simple and compound words are included in the sentiment lexicon (Devika et al., 2016; Hogenboom et al., 2013; Hogenboom et al., 2015; Kaushik & Mishra, 2014; Musto, Semeraro, & Polignano, 2014). It takes a more linguistic approach compared to machine learning, by analysing input sentence structure and semantics, without abstracting it (Hogenboom et al., 2013; Hogenboom et al., 2015).

Despite the traditional view of the lexicon-based approach to rely solely on lexical resources, in practice researchers tend to enhance it using numerous rules, that many of the resulting tools are more likely to be referred to as a lexicon enhanced rule-based sentiment analysis. Not only sentiment analysis, but lexicon enhanced rule-based method has also been used for aggression detection.

Kaushik and Mishra (2014) conducted a research on big data sentiment analysis using lexicon enhanced rule-based approach. They focused on cutting the amount of processing time without sacrificing the essential accuracy of the system itself. To attain this goal, they built their own lexicon, which was domain-specific and had every word forms to avoid stemming and utilized rules such as negation handling, emoticon handling, and hashtag extraction. They claimed that the resulting algorithm managed to process large input at a good speed and reach 73.5% accuracy.

Taboada, Brooke, Tofiloski, Voll, and Stede (2011) introduced SO-CAL or Semantic Orientation CALculator, a lexicon-based sentiment analysis tools built using a dictionary with words' polarity and strength. SO-CAL was completed with rules for intensifying words handling and negation handling. The researchers showed that a hand-crafted dictionary proved to be a very important part of the performance of a sentiment analysis tool and SO-CAL dictionary performed really well across domains.

On a research of sentiment analysis for microblog posts such as Twitter, researchers applied a splitting technique towards a single post, aiming to assess each split part better (Musto et al., 2014). Splitting was done using linguistic cues such as punctuations, adverbs, and conjunctions, and each part was assessed individually, complete with negation and intensifying words handling. Musto et al. conducted the research using four different sentiment lexicons and concluded that Multi-perspective Question Answering (MPQA) and SentiWordNet performed better compared to Senticnet and Wordnet-Affect.

Jurek, Mulvenna, and Bi (2015) researched a sentiment analysis for social media using lexicon-based approach, where they applied the algorithm to two datasets, Twitter and IMDB movie reviews. Apart from negation and intensifying words handling, the researchers also applied a normalization of each word's score and performed calculation to get the overall score. They claimed that the combination method has improved the performance of the system even though it was more efficient for short inputs such as tweet than for long documents.

Another lexicon enhanced rule-based sentiment analysis tool is VADER (Valance Aware Dictionary for sEntiment Reasoning), which had the main goal to compose a gold-standard sentiment lexicon specified for microblog inputs (Hutto & Gilbert, 2015). VADER came with its own lexicon and algorithm for sentiment analysis, which was completed with intensifiers handling (exclamation mark, capitalization, and intensifying words), contrastive conjunction ("but"), and negation. It was tested against four corpora including Twitter posts, movie reviews, technical product review, and opinion news articles and the result was compared to seven others sentiment analysis lexicons. All sentiment lexicons including VADER's are processed using VADER's rule-based algorithm. VADER outperformed other lexicons across all four corpora, even reaching a high 96% of F1 Score for social media text processing.

Hoogenboom et al. (2013; 2015) conducted two separate studies for sentiment analysis and polarity classification respectively. They performed text sentiment analysis using a lexicon-based approach with sentiment lexicon and then aggregating the result with the output from emoticon-based sentiment analysis. The most important contribution of these two research was the knowledge about how to exploit emoticons in sentiment analysis. The researchers even presented an argument that emoticon's sentiment usually dictates the whole text's sentiment. However, since emoji has been very popular lately, assessing emoticons is no longer sufficient.

In a research for aggressive text detection, Del Bosque and Garza (2014) assessed a dataset from Twitter and presented the result with a simple aggressiveness scale from 0 to 10. Del Bosque and Garza utilized a number of features in the classification technique, including document length with fuzzy rules, number of profanities, second person pronoun frequency, NS score, ANEW score, and

SentiWordNet score. NS score was a score based on the occurrence of swearwords taken from noswearing.com list, while ANEW (Affective Norms for English Words) score was derived from words that present happiness. Lastly, a sentiment score using SentiWordNet lexicon was calculated. All features were assessed individually and combined together, and it was shown that the performance of using linear regression utilizing combination of all features gave the best result.

Comparing available sentiment lexicons

Sentiment lexicon plays a very significant part in lexicon-based approach. It can be described as a ready-made dictionary of words and their respective sentiment score (Saif et al., 2016). Building a sentiment lexicon is a task that requires a large amount of lexical source, and though some research took pride in building their own lexicons, some others proved that ready-made lexicon could result in great performance as long as it is supported by an effective algorithm.

As mentioned before, lexicon-based sentiment analysis is highly dependent on its sentiment lexica, which brings forth the importance of comparing available sentiment lexica as follows:

- SentiWordNet

  SentiWordNet is, arguably, one of the most famous sentiment lexica. It has a synset structure, where words might have different senses and thus different sentiment scores. (Musto et al., 2014). SentiWordNet was built in compatibility with WordNet, a famous English lexicon that promoted an English dictionary specialized to assist natural language processing (Baccianella, Esuli, & Sebastiani, 2010; Miller, 1995).

  SentiWordNet 3.0 was compiled using semi-supervised learning and random-walk process (Baccianella et al., 2010). Due to the synset structure of the lexicon, it is suggested to be used along with Word Sense Disambiguation (Baccianella et al., 2010; Musto et al., 2014).

  SentiWordNet has been used by many types of research, both sentiment analysis and aggression detection. On a research by Musto et al. (2014), SentiWordNet showed very consistent results for both SemEval 2013 and Stanford Twitter Sentiment 2014 datasets. Hogenboom et al. (2015) used SentiWordNet 3.0 to identify the polarity of text and Del Bosque and Garza (2014) used SentiWordNet as one of the features to detect text aggression.

- MPQA Subjectivity Lexicon

This lexicon contains 8,222 words, with 4,914 negatively labelled words ranging from verbs, nouns, adjectives, adverbs, and even "anypos" (any part-of-speech) (Khoo & Johnkhan, 2017). The words are labelled with its corresponding POS tag, polarity, and intensity (Musto et al., 2014).

Even though MPQA is considered as a lexicon with a rather small coverage, in a research by Musto et al. (2014) it was found that it had comparable accuracy to SentiWordNet when used to analyse a dataset from Twitter in both SemEval 2013 and Stanford Twitter Sentiment (2014). Another experiment by Khoo and Johnkhan (2017) showed that MPQA gave a competitive result for product review sentiment analysis but unfortunately it did not perform as well for news headline sentiment analysis.

- Hu & Liu Opinion Lexicon

  This lexicon has a total of 6,790 words, where 4,783 of them are negative, Hu & Liu compiled them from customer reviews using a machine learning technique (Khoo & Johnkhan, 2017). Khoo and Johnkhan (2017) used Hu & Liu Opinion Lexicon to analyse product reviews and news headline and whilst it performed outstandingly in product review sentiment analysis, the accuracy dropped significantly when used for the latter.

- SO-CAL Lexicon

  Semantic Orientation CALculator Lexicon is a lexicon used by sentiment analysis tool SO-CAL, where the lexicon itself comprises of adjectives compiled from 400 Epinions reviews, a subset of 100 movie reviews, and positive and negative words from General Inquirer (Khoo & Johnkhan, 2017; Taboada et al., 2011). The entry of the lexicon was given a valence ranging from -5 to 5.

  When used along with its companion algorithm it was claimed to give better performance compared to other lexicons such as Google's, Maryland, General Inquirer, even SentiWordNet (Taboada et al., 2011). However, on a research by Khoo and Johnkhan (2017), where SO-CAL's lexicon was used with another algorithm instead of its own, they reported that although SO-CAL version 1.11 performed arguably well for product review dataset, it did not maintain its performance in news headline dataset.

Other than sentiment lexica, another important part of aggression detection is aggression lexica. Though not as explored as sentiment lexica, lately the interest over lexical resources to fight cyberbullying cases has been growing. Further details of some cyberbullying-related lexica are discussed below:

- Online resources

  There are many online resources aimed to support aggression detection, including noswearing.com, bannedwordlist.com, cs.cm.edu, rsdb.org, and offensive English words from macmillandictionary.com. Some of these online resources allow people to access and even contribute to the word list, such as noswearing.com and bannedwordlist.com while others are compiled by linguists and other professionals.

  On a research of corpus and lexicon for harassment research, Rezvan et al. (2018) combined all five online resources and built their own aggressive words list, divided into six categories: sexual, racial, appearance-related, intellectual, political, and others (does not belong to other five categories). The lexicon was accompanied by a manually annotated corpus of harassment text.

- Cyberbullying detection lexical database

  Power, Keane, Nolan, and Neill (2017) have started working on a cyberbullying lexicon with a more linguistic approach. The proposed dictionary will be completed with grammatical and semantic information to assist with better text aggression detection. This project is currently under development but it is very likely to contribute to better aggression detection in the future because it promises to provide a complete database that supports a more linguistical approach to aggression detection.

Supporting features

In order to develop an effective lexicon enhanced rule-based aggression analysis, there are some important features that will be discussed as follows:

- POS tagging

  Part-of-speech tagging is a process to categorize word to its part-of-speech tag (noun, verb, adverb, adjective, etc.) from a sentence (Jurafsky & Martin, 2016; Manning, 2011). POS tagging is very important in sentiment analysis because a POS tag provides much information regarding the structure of a sentence (Jurafsky & Martin, 2016), hence enabling a more linguistic approach.

  Traditionally, the tagging process took a unidirectional approach, but over the years bidirectional taggers have been known as the more successful ones (Toutanova, Klein, Manning, & Singer, 2003). Toutanova et al. (2003) however, developed a new approach to

POS tagging which was dubbed as a cyclic dependency network, which incorporated multiple features such as lexicalization, unknown word features, and smoothing.

- Stemming and lemmatization

  Stemming and lemmatization are pre-processing steps fundamental for natural language processing, both are focusing on reducing a word to its root form (Jivani, 2011). Stemming can be defined as a process to crudely chop characters from a word without considering the part-of-speech of the word itself, causing basic algorithms to wrongly reduce *improve* to *improv* (Jivani, 2011). This error may burden the sentiment analysis performance since the stemmed word might not be found in the dictionary, making the classification process less thorough.

  Lemmatization, on the other hand, works with consideration of the context of the word and its part-of-speech (Jivani, 2011). Jivani further explained how lemmatization works in a more complex way compared to stemming, by utilizing dictionary and analysing the morphological structure of the word itself. Of course, this would mean that lemmatization depends on the dictionary, if a word is not found then it will not be lemmatized.

  A research by Matsumoto, Takamura, and Okumura (2005) pointed out that the performance of sentiment analysis task would not really be dependent on whether a stemmer or a lemmatizer is being used.

- Negation handling

  One of the simplest ways to boost the performance of sentiment analysis is handling negation in a sentence. Negation very commonly happens in a sentence, e.g. “You are not pretty”. In handling negation, there are two important tasks: finding a negation point and act upon finding it.

  The most basic way to approach negation handling is by reversing the polarity of a word immediately after negation word (Jurek et al., 2015; Kaushik & Mishra, 2014). For SO-CAL (Taboada et al., 2011) and VADER (Hutto & Gilbert, 2015), however, instead of directly reversing the score, a constant is used to shift the polarity of the affected word. The process of finding negators is different for both tools, however, VADER allows up to three steps backwards only and SO-CAL performs backwards search until a boundary is found.

- Intensifiers handling
    - Intensifier words

      Intensifier words are words used to intensify another word in a sentence, such as *very, quite, most* (Jurek et al., 2015). Meanwhile, Quirk et al. differentiated intensifier words into two main categories: amplifiers (*very*, *most*) and downtoners such as *slightly* (Taboada et al., 2011). Essentially, intensifier words are handled by simple addition or subtraction, without considering the intensity of the intensifiers (Kennedy and Inkpen, 2006; Polanyi and Zaenen, 2006, as cited in Taboada et al., 2011).

      Jurek et al. (2015) compiled a list of commonly used intensifiers and categorised them into three categories namely downtoners, weak amplifiers, and strong amplifiers. Then a percentage was assigned for each category, -50%, +50%, and +100% respectively. Additionally, a rule of neutrality was applied, where intensifier words are regarded as neutral when surrounded by neutral words.

      SO-CAL took it even further by assigning modifier percentage to each intensifier word in the dictionary (Taboada et al., 2011). The researchers also took into account adjectival intensifiers or adjectives that intensify nouns in phrases such as “big problem” and “total disaster”, and they had a separate intensifier words dictionary for this.

    - Punctuation

      Certain punctuations have also been known as intensifiers, such as exclamation mark and question mark. Hutto and Gilbert (2015) reported that exclamation mark intensified sentiment score since the sentence “I love it!!!” was considered to be more intense compared to “I love it.” This intensification would make a negative statement more negative and positive statement more positive, meaning it would not shift the polarity of the statement.

    - Capitalization

      Instinctively, the reader might consider capitalization as an emphasis on a certain point. Therefore, capitalization has served as an intensifier in some research. In VADER, when a word is written in ALL-CAPS, the sentiment score would be intensified (again, without shifting the polarity) by an empirically derived constant (Hutto & Gilbert, 2015). They established baseline sentences and then made variations in capitalization on sentiment laden words in the sentences such as “bad” to “BAD” and asked manual workers to give

sentiment score for both baseline and test condition sentences. Based on the sentiment score they calculated the mean value of the difference between baseline and conditioned sentence and came up with a scalar of 0.733 for ALL-CAPS modifier.

In a research of cyberbullying detection using rule-based method, however, excessive presence of capital letters (more than 50%), the whole text would be marked as cyberbullying because Bayzick et al. (2018) argued that excessive use of capital letters was an indication of hostile communication.

2.3 Emoji and emoticon sentiment

A study on harassment detection has shown that manual human labour of detecting aggression might also be affected by ambiguity, given that some of the posts are actually jokes between friends (Yin et al., 2009). Considering the usage of emoticon and emoji as replacements for non-verbal cues in online communication, they can also be used to disambiguate the context of a text.

Past two research about emoticons by Hoogenboom et al. (2013; 2015) have found that people tend to use emoticons to convey their real feelings, hence the sentiment of emoticon would generally dominate over text sentiment of a post. They compiled an emoticon sentiment lexicon, which was further used to assist in sentiment analysis and text polarity classification tasks. In both of research, it was presented that emoticon actually helped to improve the performance. However, considering the rising popularity of emoji (Marengo et al., 2017), research on emoticon is no longer sufficient.

One of the most vital thing that differentiates emoji from emoticon is that emoji is represented in Unicode format, meaning it has a universal standard. This makes identifying emoji easier compared to emoticon because emoticon does not really have a standard and could sometimes be inconsistent. Emoji, on the other hand, have a standardized list of Unicode representations[1].

A study classifying emoji's sentiment was conducted by Kralj Novak et al. (2015) which focused on compiling the first emoji sentiment lexicon named Emoji Sentiment Ranking. They collected a huge number of tweets with emojis in 13 different languages and arranged native speakers as human annotators to each corresponding language to label the tweets. Then the score of tweets was aggregated to individual emoji sentiment score. It was also concluded that emoji is language-independent, meaning there is no significant gap in usage between languages.

[1] unicode.org/emoji/charts/full-emoji-list.html

2.4 Challenges of processing text from social media

Social media is the new platform for human behavioural studies, laden with personal data and opinions. However, it is also important to realize the reliability of social media data, of how it could be distorted and bias, and if there is a workaround solution to overcome the limitations. The limitations or gaps of past research can be grouped as follows:

2.4.1 Linguistical limitations

Social media is not a formal communication media, hence it is impossible to expect formal and proper language use in communicating via social media. Misspelt words, elongated words, slangs, and grammatical errors are a few examples (Desai & Narvekar, 2015).

Misspelt words could be both intentional and unintentional, such as people trying to shorten words like “you” to “u”, or unintentionally misspelt “cake” to “cske”. Elongated words are words with excessive characters usually used to emphasize a point, compare “I like you” to “I likeeeeee you” as an example. Slang is informal words used in informal communication but are not usually present in vocabularies, it might include abbreviations such as “lol” and “rofl”, or shortened words like “b4” (“before”). These uncommon words are referred to as NIV (not-in-vocabulary) words by Desai and Narvekar (2015). The researchers presented a method to normalize these words, starting with identifying NIV words using PyEnchant Library of Python, then processing the NIV words using a combination of word shortening algorithm, slang replacement, and lexical matching with Levenshtein Distance. Desai and Narvekar noted that this normalization could contribute to sentiment analysis task since proper English is needed for better sentiment analysis performance.

2.4.2 Non-linguistical limitations

Social media offers an experience of freedom of speech, with anonymity as the spearhead. There is usually no background check when registering for social media, anybody could easily pose as someone else, make as many social media accounts as possible, and get away with it. This freedom brings forth issues when social media is used as behavioural study media. Anonymity brings the worst of human since people think they would not be found when they did something wrong (Ruths & Pfeffer, 2014).

When assessing a certain scope using social media, such as a study of the level of government satisfaction of Canterbury residents, the data could be biased. While some people might just put Canterbury as their address even when they are not actually living there, others might actually live in Canterbury but did not want to put the information in their account. Cohen and Ruths argued that this issue would affect the reliability of data collected from social media (Ruths & Pfeffer, 2014).

Another issue arises from the fact that social media has different forms and standards. Just how Facebook is different from Twitter, research focused on assessing Facebook data is usually not compatible with Twitter data. Ruths and Pfeffer (2014) presented an argument regarding how research on a specific social media platform usually suffers from overfitting, meaning it would work efficiently on the corresponding social media only, while the performance would drop significantly when used in other platforms.

2.5 Inter-annotator agreement

In building a labelled dataset for training and testing purposes, manual annotation by human annotator is needed. In a study of cyberbullying detection using Formspring.me data, Reynolds et al. (2011) utilized three annotators to label the data, and a text was only considered bullying when two or more annotator agreed on it. The annotators were hired anonymously via Amazon's Mechanical Turk, where it claimed to be giving a fast result even with the high amount of data. However, the validity of the annotators is in question because there is no specified inter-annotator agreement.

Annotation is usually done with more than one annotator, and the validity of the annotation done by two annotators could be calculated using Cohen's kappa (Bobicev & Sokolova, 2017). The formula is as follows:

$$\kappa = \frac{P_o - P_e}{1 - P_e} = 1 - \frac{1 - P_o}{1 - P_e}$$

Where P_o is the observed agreement between annotators, and P_e is the probability of random agreement, which can be calculated as follows:

$$P_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

P_e can be defined as the sum of probability that both annotators agree at random or expected agreement, where N^2 is the total number of entries annotated, k is the total number of annotation possibility, and n_{ki} is the number of times annotator i marked entries as category k.

Landis and Koch (1977) presented a guideline to interpreting kappa value, where 0.41-0.60 is deemed moderate, 0.61-0.80 as substantial, and above 0.81 as almost perfect. It was also pointed out that this guideline was compiled based on their personal opinion, as it was argued that there is no universal guideline to kappa value (Bakeman, McArthur, Quera, & Robinson, 1997; Sim & C Wright, 2005).

Chapter 3

Method

This chapter focuses on discussing the research method used in the study in order to achieve its objectives: identifying elements of cyber aggression in social media and evaluating the algorithm's performance.

3.1 Experimental Method

This section outlines the methods used to identify elements of cyber aggression, starting from choosing, collecting, and cleaning obtained data and finally developing the data processing algorithm.

3.1.1 Obtaining Datasets

Online social media has been developing rapidly in recent years, resulting in variations of online communication form. Existing research towards cyberbullying or cyber aggression detection usually focused on only one communication form, making the research result inapplicable to other forms of communication. In this research, three most common online textual communication styles are assessed: single post style (will be referred to as "single style"), question and answer (QA), and thread style. All datasets selected are in English.

Single Style

Single style means each input is treated as a single entity, like a single Facebook comment or status post or a single Twitter tweet. This is seen as the most basic out of the three assessed styles, where the input is assessed without considering the neighbouring posts. The single style dataset chosen was comments taken from Melania Trump's official Facebook page It was chosen because Facebook has been considered as one of the most widely-used social media and Melania Trump, being the wife of the current president of the USA, has received a lot of hate from citizens of the United States, who are mostly English speakers.

Extraction of comments was done using a Python program utilizing Facebook's Graph API and writing the results into a CSV file. There are many ready-made Facebook scraping programs that are

available such as the one shared by Paulo (2017), even though several modifications were needed to ensure that the captured emojis are in the right Unicode representation and to fit the changes in the API itself. The correct Unicode representation is needed in order to be able to associate it with emoji sentiment lexicon used in the research which is Emoji Sentiment Ranking by Kralj Novak et al. (2015).

Facebook's API itself has been through many changes, including a substantial change regarding security in August 2018. Gaining access to the API used to be done by simply applying for a key through Facebook App page as an app developer, but the latest change to permission and access requires developers to go through stricter verification process. This security measure, though necessary to protect Facebook users' data, unfortunately also limits data scientists' access. All data used in this research were collected in April 2018, when the API's access was still open.

After extracting the data, refinement was needed to remove irrelevant data such as advertisements, news, spams, non-English entries, and duplications. Overly long texts were also removed because they are usually irrelevant and single style processing is focused on shorter texts. To build training dataset, 202 texts were randomly selected and then annotated. This training dataset consists of 141 aggressive texts and 61 non-aggressive texts. The remaining data were then shortlisted as a testing dataset to 403 texts to speed up manual annotation process. Manual annotation was done by two native English speakers annotators and evaluated using Cohen's kappa (will be discussed in Section 4.2.1). Since the annotation was done manually by human, the number of entries is limited to make sure annotation can be completed in one session to make sure they maintained the same perception about cyberbullying while annotating.

For an entry to be included in the final testing single style dataset, it has to be on mutual agreement of both annotators. Out of 403 entries, 356 entries were agreed upon while 47 entries with different labels from annotators were removed. The dataset itself is unbalanced, with 210 entries marked as aggression and the remaining 146 were marked as non-aggression.

For the other two styles – QA and thread, existing datasets used in previous studies were chosen after assessing other available annotated datasets (Golbeck et al., 2017; Xu, Jun, Zhu, & Bellmore, 2012).

QA Style

For QA style, this research utilizes an existing labelled dataset from Reynolds et al. (2011), where pairs of question and answer were taken from Formspring.me, a social media considered having a high level of aggression. From their dataset, a smaller dataset of 1,000 entries was selected as testing dataset for this research. It consists of 500 aggression entries and 500 non-aggression entries.

Another dataset for training purpose was created from 402 QA entries, with balanced number of aggression and non-aggression.

The result from past research showed that the rule-based model managed to reach recall percentage of 78.5% (Reynolds et al., 2011). They also argued that in aggression detection, recall is more important than precision. This means that the main concern of the research is to correctly identify aggression, even with the cost of falsely labelling non-aggressive entries as aggression.

Thread Style

The dataset for thread style was taken from a study by Bayzick et al. (2018). They selected MySpace.com as their source, where they grouped together connected entries in a single "window" or thread. There are eleven packets, each packet contains 130 to 230 windows. Each window was annotated manually by three human annotators and a window would be marked as bullying when two or more annotators marked it as such.

For this research, the first packet was chosen as training dataset while the second, third, and fourth packets were used for testing purpose. All packets are unbalanced, with packet four containing only non-aggression. The packet was chosen specifically to check the resulting algorithm's ability to correctly identify true positives.

Bayzick et al. (2018) showed that the accuracy of the final software varies from 32.32% to 83.97% for each packet. Overall they managed to identify 354 out of 415 cyberbullying windows and 855 out of 1647 of non-aggressive windows. Using the number in a calculation, the recall and precision of the software is 85.3% and 51.9% respectively.

All three sources of datasets (Facebook, Formspring.me, and MySpace.com) have been considered as social media with high level of aggression (Bayzick et al., 2018; Reynolds et al., 2011; Whittaker & Kowalski, 2014), which means they would provide better understanding of identifying elements of cyber aggression in different platforms of social media.

3.1.2 Data Processing Algorithm

In this section, algorithm process flow and its details are discussed. In order to answer the research questions, lexicon enhanced rule-based method is chosen, utilizing various lexica starting from swearwords lexicon, SentiWordNet, slang dictionary, positive word lexicon, emoticon lexicon, and Emoji Sentiment Ranking. This approach has been considered to be a robust approach to detect aggression in texts, because of the flexibility of combination of rules that can be applied (Bayzick et

al., 2018; Del Bosque & Garza, 2014; Reynolds et al., 2011). As discussed before, this approach takes a semantic approach, by considering linguistical elements and thus will give better insights on identifying elements of cyber aggression in textual communication.

Generally, the classification process is done in three to four steps, starting with cleaning up text data, processing single data, classifying single data, and additional classification when the data type is either QA or thread style. Every type of inputs will follow step one to three, while QA and thread inputs will be processed further according to its type. This is done with the aim of generalization and reusability of the algorithm. Figure 3.1 depicts the general flow of the algorithm.

In this study, a QA input is made of paired single input while thread style consists of up to ten single inputs. Every single input consists of one or more sentences and each sentence consists of one or more words.

The resulting software is written in Python 3 and enhanced by Python's NLTK and some other relevant Python's libraries which will be discussed further in the next section. The software can be called using a normal Windows PowerShell or Terminal, with an additional parameter to specify the input type, whether it is a QA or thread style. Not specifying input type will result in a single type input mode. The input for a single type and QA type would be a CSV file name, while thread style would ask for a folder name with CSV files inside it. The result of the software will then be written in an XML file for readability.

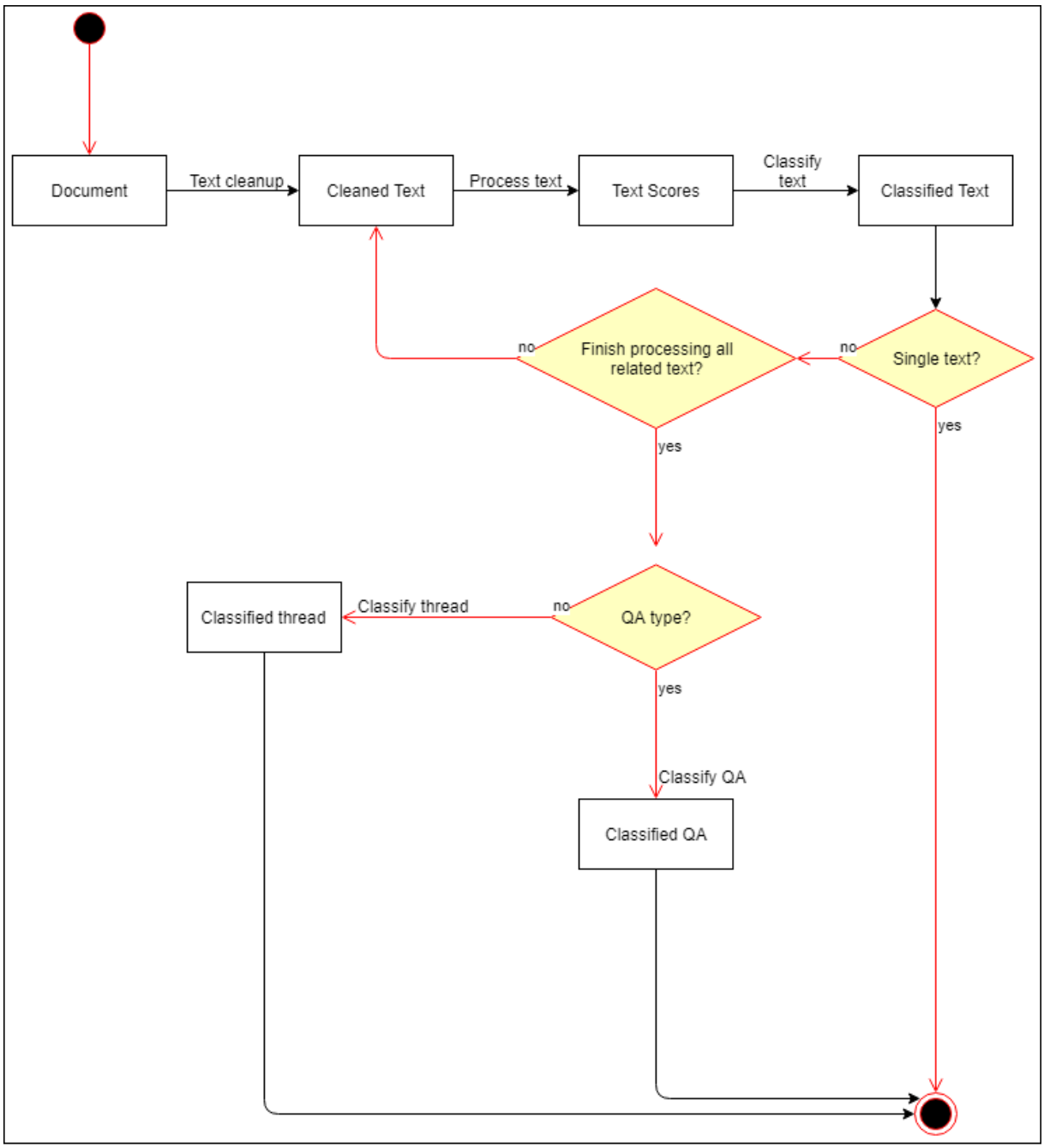

Figure 3.1 General flow algorithm

Text Cleanup

The first step of cleaning up stage is separating emojis from plain texts. This is done by utilizing the Unicode of emojis from emoji lexicon and regular expression split technique. The remaining text will be processed further to extract emoticons using the same technique with emoji extraction, using emoticon lexicon instead of emoji. QA style input has to be split into pairs first before processing each part using the same cleaning and processing technique. The simplified text cleanup algorithm is shown in Fig 3.2.

```
BEGIN text_cleanup

  sub slang_handler(text):
    return text


  sub process_niv(token):
    suggested_token = PyEnchant suggested word
    return token from suggested_token with best distance score


  sub capital_percentage(tokens):
    return capital_percentage


  sub remove_stopwords(pos_tag, tokens)
    return pos_tag, tokens


  sub clean_text(text):
    return pos_tag, tokens, capital_percentage, punctuation_count


  sub emoticon_handler(text):
    return emoticon_sentiment, new_text


  for each r in inputs:

    if input type is single text or thread:
      emoticon_sentiment, text_only = emoticon_handler(text)
      text_details = clean_text(text)

    else if input type is QA:
      for each text in text_array:
        text, emoji = separate emoji from text

        emoticon_sentiment, text_only = emoticon_handler(text)

        text_details = clean_text(text)

      pair text_details of Q and A together

  return whole cleaned up document

END
```

Figure 3.2 Text clean up pseudocode

The example below shows an input before and after being separated:

> In: "\\U0001f1fa LOL every1 h8 ur ugllly face. I kid you not, idiot~. u better kill urself:-)"
>
> Out: [["LOL every1 h8 ur ugllly face.", "I kid you not, idiot~.", "u better kill urself"], ["\\U0001f1fa"], [":-)"]]

After separating plain text from emoji and emoticon, the text is cleaned further. Punctuation, which has been considered as an intensifier, is counted and this value is kept in a variable for future processing. Then non-boundary punctuations is removed from the text. Non-boundary punctuations are punctuations that are not considered to be a grammatical boundary such as a comma, semicolon, colon, apostrophe, and so on. Boundary punctuations are not removed because they can serve as a boundary when processing the text further. The idea of using boundary punctuations is adapted from SO-CAL (Taboada et al., 2011). See example (1) below for details on boundary punctuation in sentences.

(1) a. What a ***very handsome***, smart, and funny guy.

b. I think you are ***not ugly***; stupid, yes.

c. I ***kid*** you ***not***, idiot.

In example (a) above, the modifier "very" supposedly only affects the word "handsome" and will not have any effect on "smart" and "funny". While this case might only affect the affinity of a sentiment, boundary punctuations have more impact on negation handling as in examples (b) and (c), where without considering the semi-colon and comma, the negative word ("stupid", "idiot") would be negated.

Unlike formal texts such as newspaper or official reviews, social media interaction is not grammatically reliable. This unreliability or "noisy" texts need to be normalized properly if a more semantical approach of text processing is going to be applied. Text normalization is done in five steps: pronoun spelling resolution, slang resolution, laughter text resolution, elongated characters reduction, and similar word replacement.

One of the findings of this research is that a lot of past research usually overlooked resolving pronoun spelling. In informal texts users are usually too lazy to write pronoun properly, often shortening it from "you" to "u" or even omitting important apostrophe like "I'm" to "im". If this issue is not addressed properly, it will be a liability when applying POS tagging and further semantical processes. To resolve this issue, a replacement algorithm is implemented to resolve common pronoun spelling mistake.

(2) a. “U better kill urself” -> “You better kill yourself”

b. “shes too ugly to be a model” -> “She’s too ugly to be a model”

It may seem simple, but based on experiments during the development period, a missing apostrophe in a pronoun might result to a faulty processing in further stages such as falsely replacing “urself” with “ourself”, which is no longer a second or third person pronoun. As discussed in Chapter 2 (Literature Review), one of the most important indicators of aggression is the presence of second person pronouns (Bayzick et al., 2018; Yin et al., 2009). The importance of correct pronoun in aggression detection will be discussed further in later section of Text Processing.

The next step of noisy text normalization is to resolve known slang using a list of slang and its corrected entry. The slang dictionary is built using a list from the website noslang.com. If a slang phrase is found in the slang lexicon, it is replaced by the corrected form. For example (3b), the word LOL was replaced with “laughing out loud”, “every1” to “everyone”, and “h8” to “hate”. After resolving found slang, the text is tokenized and each word is assessed for validity. NIV or invalid words is processed further using the last three normalization steps. The bolded words in example (3) are examples of NIV words that will be processed further using NIV cleaning steps.

(3) a. You are so ***stypid hhahahahhaaa***.

b. LOL ***every1 h8*** your ***ugllly*** face.

Laughter text resolution is a task of recognizing laughter text and converting it into a simpler, easily recognizable form. For this research, the accepted laughter text is variations of “haha”, it could be as simple as “hahahaha” or as elongated as “hhaahahhhaahha”. Detection is done using a regular expression and if a word is considered as a laughter text, it is not going to be normalized further and stored as a laughter cue “haha”. Laughter cue serves as possible neutralizer when classifying an input. In example (3a) above, the elongated laughter is shortened into a simple “haha”.

Removing elongated characters in a word is done by utilizing regular expression function, reducing repeated characters to a maximum of two repetitions only. The most optimum example is reducing “happppy” to “happy” which is a valid English word. In another case, the reduction might not be too successful, such as reducing “haaappyyyy” to “haappyy” or even invalid words without elongated characters such as “cske”, which are still not valid English words. In a case of invalid word even after characters reduction, word replacement process is needed. For example, in (3b) the word “ugllly” is shortened to “ugly”, which is still an NIV but simplified enough for more accurate word replacement in the next step.

Word replacement process is a task of finding the most similar word as a replacement for NIV or invalid word. PyEnchant Library provides a function that gives suggestions of similar words that can be used to replace an invalid English word. Then the distance between each suggested word and the invalid word is calculated using FuzzyWuzzy Library's ratio. Then based on the distance score, the suggested word with the best score is chosen as the replacement word. For example, the word "uglly" from shortening algorithm got two suggested words by PyEnchant: "gully" and "ugly" and using FuzzyWuzzy's ratio function, "gully" got 80 and "ugly" got 89. As the maximum ratio is chosen as replacement word, "ugly" was chosen to replace "uglly".

By the end of NIV cleaning stage, the sentences from example (3) would become:

a. You are so stupid haha.

b. Laughing out loud everyone hate your ugly face.

After normalization, the text is POS tagged using NLTK POS tagger, to make sure that in future stages, each token can be processed properly. One English word can have more than one POS tag, which will affect the sentiment and context of the whole text. Then stop words removal is done using a list of modified stop words from NLTK (pronouns and negations are removed from the stopwords list). Stop words removal is done after POS tagging to ensure that POS tag is as grammatically accurate as possible, otherwise, some words might be removed beforehand and the sentence is no longer sensible.

The last outcome after POS tagging would give the sentences below:

- [('Laughing', 'VBG'), ('loud', 'JJ'), ('everyone', 'NN'), ('hate', 'VB'), ('your', 'PRP$'), ('ugly', 'JJ'), ('face', 'NN'), ('.', '.')]
- [('I', 'PRP'), ('kid', 'VBP'), ('you', 'PRP'), ('not', 'RB'), (',', ','), ('idiot', 'NN'), ('.', '.')]
- [('You', 'PRP'), ('better', 'JJR'), ('kill', 'VB'), ('yourself', 'PRP')]

Text Processing

After cleaning process, separated emoji, emoticons, and text is processed further for different scores: emoji sentiment score, emoticon sentiment score, text sentiment score, text aggression score, and text positive term score. The goal of text processing stage is to produce scores that will be used in classification stage later on. The output scores will be forwarded for all three classification processes:

single style, QA style, and thread style. Separating scoring and classification is done to promote reusability, where the scores can be used in different approaches of classification methods.

While emoticon and emoji scores are calculated once for every single input, text scores are calculated for each sentence. If a single input consists of three sentences then it will have three aggression scores, three sentiment scores, and three positive word score, each corresponds to each sentence respectively. These scores will then be forwarded for determining whether an input is aggression or not in classification stage. Fig 3.3 shows the simplified pseudocode for text processing, while the full version can be found in the appendix. In this stage several features such as modifier application, but check, and least check is utilized. More details regarding these features will be discussed further in later sections.

```
BEGIN process_text

   sub apply_modifier(index, score):
      return score

   sub but_check(sentence_score):
      return new_score

   sub get_sentiscore(text):
      sentence_score = but_check(sentence_score)
      return average sentence_score

   sub check_aggression(token, text):
      return if aggression

   sub get_aggression_score(text):
      sentence_score = but_check(sentence_score)
      aggression_score = sum of sentence_score

      if any laughter present:
         aggression_score = aggression_score + 1
      return aggression_score

   sub get_positive_score(text):
      sentence_score = but_check(sentence_score)
      return sum of sentence_score

   sentiscore = get_sentiscore(text)
   aggression_score = get_aggression_score(text)
   positive_score = get_positive_score(text)
   emoji_sentiment = emoji_sentiment_analysis(emoji)

   return scores
END
```

Figure 3.3 Text processing pseudocode

Emoji Sentiment Score

Emoji sentiment analysis is quite straightforward because emoji is standardized and has its own Unicode representation. The lexicon chosen is Emoji Sentiment Ranking, where each emoji, its Unicode, and sentiment score is listed. Sentiment score ranges from -1 to 1, the same with the software's standard scoring range, so no scaling is needed.

Emoji Sentiment Ranking listed 969 emoji complete with its number of occurrences (divided into positive, neutral, and negative occurrences) and after filtering irrelevant emojis the number went down to 467. The filters applied are as follows:

- Occurrences less than 5
- Sentiment score of 0.000
- Irrelevant emoji groups such as arrows, block elements, box drawing, geometric shapes, etc.
- Emojis with positive and negative occurrence less than 0.4 or neutral emoji. This specific rule was not applied to emoticon emoji, which are emojis representing human emotions.

If one emoji appears more than once in a single input, its sentiment score will be increased/decreased by 10% every time it reoccurs. This is based on the assumption that when the same emoji is being used more than once, the originator might be trying to convey a stronger feeling, a concept similar with using ALLCAPS in text, which requires extra efforts while the originator was typing. Thus, the formula used to calculate each emoji's overall sentiment is as follows:

$$S_{(e_m)} = \sum_{n=1}^{m} \left(S_{(e)} \times \sum_{a=1}^{n} 10^{1-a} \right)$$

Where:

- $S_{(e_m)}$: Aggregated sentiment of multiple occurrences m of emoji e in an input
- $S_{(e)}$: Sentiment score of emoji e taken from Emoji Sentiment Ranking
- m : the number of occurrences of emoji e in an input
- a, n : counter for summation

If there is more than one emoji in a single input, firstly each emoji is aggregated using the formula above (in case of reoccurrences of each emoji) and then it is summed to get the emoji's whole sentiment. When an emoji is not found in the lexicon, it is not included in the calculation.

For example, there is an input with three "angry face" and one "unamused face", both can be found in Emoji Sentiment Ranking with sentiment score of -0.302 and -0.375 respectively. Using the formula above on each emoji gives the result of -0.96942 for "angry face". Thus the overall emoji score would be -0.96942 + -0.375 = -1.34442.

Emoticon Sentiment Score

Calculating emoticon sentiment score takes a simpler approach compared to emoji sentiment calculation. Because of the needs to type emoticon manually, it is not common to repeat emoticon to emphasize a feeling, thus the more complex way of subtly enhancing sentiment score is not applied when calculating emoticon sentiment score.

Emoticon sentiment score is calculated as the sum of sentiment score of emoticon that appears in a single input. The sentiment score is obtained from emoticon sentiment lexicon by Hogenboom et al. (2013), where the scores are either -1, 0, or 1. In a case that an emoticon is not listed in the lexicon, it will be disregarded.

Aggression Score

Aggression score is calculated using swearword lexicon of 1,717 swear words. The swearword lexicon used in this research was composed of different sources: noswearing.com, bannedwordlist.com, cs.cmu.edu, rsdb.org, and bad words list from cyberbullying research by Engman (2016). The list was updated manually during the research period for words that are deemed too ambiguous to be listed as swearwords such as American, adult, Africa, Asian, bigger, taboo, toilet, and so on. These words were removed because they raised too many false positives and cannot be considered as aggression if they are not accompanied by explicit insult words.

Each word is lemmatized using NLTK lemmatizer and passed to an aggression check function. Only noun aggressive words are directly returned as aggression while other POS is checked for presence of second or third person pronoun. This was done because noun is self-explanatory. On the other hand, other POS words are usually preceded or followed by noun or pronoun. See example below for more details:

a. Noun

   There are a lot of cases when a sentence only contains a single word of insult, most of the time it is a noun such as “idiot”, “b*tch”, or “c*nt”. There is no need to further check these sentences for presence of pronoun because those words can only refer to human being and may or may not be accompanied by pronoun or noun.

b. Adjective

   Adjectives are usually used to explain nouns, such in sentences like “***You’re*** so ***stupid***” and “What an ***ugly girl***”. It is necessary to check for second or third person pronoun or noun to make sure that the adjective was meant as an aggression, otherwise cases like “***I’m stupid***, sorry” will be falsely detected as aggression.

c. Verb

   Similar to adjectives, verbs are meant to be accompanied by pronoun as a subject or object. Pronoun checking is added to properly distinguish different contexts such as in “We all ***hate you***” and “Love and ***hate*** is part of life”.

If a word is marked as aggression it will be given a score of -1 which can be modified during modifier handler function. Modifier handler function is a common function used when calculating aggression score, positive word score, and sentiment score. More details regarding this function will be explained in Modifier Handler Function section.

Overall sentence aggression score is calculated as a sum of aggression score of each word, with laughter cue contributes to +1 score as a possible neutralizer. If no aggressive word is found but a laughter cue is present, the overall sentence aggression score will be +1.

Positive Word Score

Positive words are words that have positive connotations such as radiant, able, agree, prosper, etc. There is a total of 710 positive words in the list, taken from a cyberbullying detection research by Engman (2016). Each lemmatized positive word found gets a +1 score, which also can be modified using modifier handler function. Like aggression score, the sentence overall positive word score is the sum of positive score of each word.

Sentiment Score / Sentiscore

Sentiment score or sentiscore is calculated using SentiWordNet as the sentiment lexicon. Just like aggression score and positive word score calculation process, each word is lemmatized with NLTK lemmatizer first before it is processed further for sentiscore calculation.

As discussed earlier in Chapter 2 (Literature Review), SentiWordNet was built using a synset structure, meaning each word can have one or more synsets or word sense. Since the objective of this research is to identify elements of cyber aggression, the synset with the most negative sentiment score is selected during sentiment score calculation.

Then each word's sentiment score will be processed further for possible modification through modifier handler function and aggregated as overall sentence sentiscore.

Modifier Handler Function

Modifier handler function is a function used during all three text scores calculation and is the most important part of text processing stage. It is separated from scoring calculation function to promote reusability. This function utilizes capitalization, modifier words, negation handling, "least" check, and "but" check.

1. Capitalization has always been considered as sentiment intensifier, meaning excessive capitalization is amplifying conveyed emotion (Bayzick et al., 2018; Nalinipriya & Asswini, 2015). In past researches, only all caps words are intensified but in this research, a word with 50% or more capitalization is considered amplified. Example (1) shows comparisons of capitalization usage where it can be seen that even though the word "ugly" is not written in all caps, it still somewhat emphasized the insult given.

    (1) a. You are so ugly

    b. You are so UgLy

    c. You are so UGLY

    It can be concluded that when excessive capitalization is used, the sentiment will be amplified by a predetermined scalar. This scalar was empirically derived and used in VADER research by Hutto and Gilbert (2015). Hutto and Gilbert calculated the scalar by comparing sentiment intensity from grammatical and syntactical cues and came up with a scalar of 0.733 used for amplifying sentiment-laden words with ALL-CAPS variation. In this pseudocode, however, sentiment-laden words with 50% or more capitalization (instead of

100%) will be amplified using VADER's scalar. Amplification is done without disrupting the polarity of the sentiment, meaning it will make the positive score more positive and vice versa.

2. Modifier words

Modifier or intensifier words are words that can cause shifts in the corresponding word's sentiment. For this research, modifier words are divided into two categories of increment and decrement words. Hutto and Gilbert (2015) referred to these words as booster words. Like capitalization handling, any found modifier word will increase or decrease the score of the affected word(s) by an empirically derived scalar taken from VADER. In this case, the scalars were calculated as +0.293 and -0.293.

In VADER modifier word search was done statically to up to three backwards steps, but for this research, a more flexible approach was chosen. Backwards steps are done until a boundary is found, a boundary could be a word or punctuation, as discussed before during clean up stage section. This more flexible approach was inspired by SO-CAL (Taboada et al., 2011).

Combining approaches used by VADER and SO-CAL, the further a modifier word is from the affected word, the least effect it will have on the word's score. Example (2) shows the modifier word "very" affecting the word "ugly". If the modifier word is two steps behind the affected word, the scalar will be reduced by 5% (a), three or more steps will reduce the scalar by 10% (b).

(2) a. You are ***very*** bad ***ugly***.

b. You are ***very*** bad stupid and ***ugly***.

3. Negation handling

In detecting aggression, calculating sentiment, and detecting positive words, negation is a very important part that can change the whole approach (Hutto & Gilbert, 2015; Jurek et al., 2015; Taboada et al., 2011). Comparing several negation handlers in past researches, VADER's approach was selected as the most suitable approach. Negators are compiled as words that negate the following word. Other than checking from a list of negators, a checker for token "n't" is also added.

The list of negators was acquired from VADER as well as the modifying scalar for negation shift. In this case, the scalar is defined as -0.74, but instead of adding or subtracting the score

with the scalar, in negation handling the score is multiplied by negation scalar. Multiplication is selected instead of addition because there is a possibility of overly positive or negative words that cannot be negated with scalar or -0.74. Consider example (3a) where the word "MARVELOUS", which is really positive and amplified with capitalization might have an overly positive score of, for example, 0.8. Even after negated with scalar above, it will still have a score of 0.06, which is not correct. Another reason is that each swearword found will be given a score of -1 and in cases like example (3b), it will result in a score of -0.26, which will also be incorrect. By using multiplication, it is certain that the polarity will be shifted.

(3) a. You are not MARVELOUS at all.

b. You are not a c*nt.

While searching backwards until meeting a certain boundary is applicable for modifier words handling, in searching negation it does not seem correct to do so. Consider example (3), where (a) is a direct, more practical negation while (b) is something that is more likely to be encountered in a social media text communication: lacking in grammatical cues. In case of (b), since the word "type" is not really an insult, the word associated with negator "not" would be "ugly" and negating would make it seems like an innocent sentence. To prevent false negation handling, the backwards count is limited to a maximum of three steps or until it encounters a boundary. By applying this limitation in the backwards count, negation will not be done for example (b).

(4) a. You are ***not beautiful***.

b. Wow, you are ***not*** my type ***ugly***.

4. "Least" check

In a sense, checking for word "least" in a sentence can also be considered as negation handling but since it was handled differently in this research, it will be discussed separately. The approach chosen is derived from VADER's (Hutto & Gilbert, 2015) approach. When encountering the word "least", another one step backwards is taken to check if it was preceded by "at" or "very", if yes then it will not be considered negation. Example (5) illustrates the differences between sentences with "least" with and without "at" or "very". By applying this rule, (a) would be negated while (b) would not be negated.

(5) a. You are the ***least ugly*** girl here.

b. He is ***at least*** as **handsome** as Jesse.

5. “But” check

   “But” check is another grammatical approach of text sentiment analysis task inspired by VADER (Hutto & Gilbert, 2015). They showed that it is a simple yet powerful and syntactically correct way to approach the word “but”, by shifting the words’ polarity before and after finding the word “but”. If “but” is found in a sentence, all words before it will have its score halved while words after it will be multiplied by 1.5. For example, in (6) it can be seen that the emphasize should have been put on the phrase after the word but.

   (6) You are bad, but ***I don’t hate you at all***.

Single Style Classification

As discussed earlier, a single input consists of one or more sentences and each sentence has its own text scores (aggression score, positive word score, and sentiscore). Single style classification stage combines all scores including emoticon and emoji scores to come up with the single input classification. Since QA and thread style inputs actually consist of one or more single style inputs, all three input types will go through this stage before being processed further according to its type. Fig 3.4 shows the pseudocode for single style classification, where scores generated from Text Processing stage are used to determine whether a single input is considered aggression or not by establishing a set of rules that will be discussed further below.

```
BEGIN classify_text

  for text, emoji, emoticon_sentiment in full_document:

    sentiment_score, aggression_score, positive_score = process_text(text)
    emoji_sentiment = emoji_sentiment_analysis(emoji)

    sentiscore_benchmark = average of sentiscore

    for each sentence scores:
      if aggression_score < 0:
        if aggression_score + positive_score > 0:
          overall_score = aggression_score + positive_score
          if sentiscore > 0:
            overall_score = overall_score + sentiscore
        else if
          overall_score = aggression_score + positive_score + 1

          if sentiscore < 0 and sentiscore <= sentiscore_benchmark:
            overall_score = overall_score + sentiscore
      else if positive_score > 0:
        overall_score = positive_score
      else if sentiscore < 0 and sentiscore <= sentiscore_benchmark:
        overall_score = sentiscore
      else:
        overall_score = 0

      if overall_score < 0:
        overall_score = overall_score - (punctuation_constant) * punctuation_count
      else:
        overall_score = overall_score + (punctuation_constant) * punctuation_count

      push overall_score of each sentence to array

    if any sentence has aggression_score < 0:
      if sum of negative aggression_score + emoticon_sentiment + emoji_sentiment < 0:
        text is aggression
      else:
        text is not aggression
    else:
      text is not aggression

END
```

Figure 3.4 Single text classification pseudocode

To answer the research questions, this is the stage where a lot of experiments were conducted.

Various possible combinations of score calculations were implemented such as:

1. Weighted calculation

   This is a method where each score is assigned with a weighted value that will make up a full score of 1. Some formulas were used during the research to find the best fit. After comparing several combinations, one of the best performing formula is:

$$Overall\ score = aggression\_score * 0.4 + sentiscore * 0.2 + pos\_word\_score * 0.2 + emoji\_score * 0.1 + emoticon\_score * 0.1$$

   Note that the example above is just one of many possible combinations of weights in the weighted calculation. The above formula was built under the assumption that text scores are more important than emoji and emoticon scores, and text aggression score holds the highest importance in getting overall aggression score. Also worth noting is that not every text has emoji and emoticons, hence the most prominent feature to assess aggression is the text itself.

   While seemingly fair and instinctive, assigning proper weight is a challenge and might not be backed up by empirical evidence. As stated above, all possible weighing formulas this research have been experimented on were based on assumptions only and did not give stable results. There have been very few studies that applied weighing formula in sentiment analysis or aggression detection, hence there is no supporting information that can be used to justify which feature is actually the most important.

2. Single score calculation

   Single score calculation means using only one score as an indicator of aggression in a text. In this case, the most obvious score to be chosen is, of course, text aggression score.

3. Linear combination calculation

   This calculation is taking a filter-like approach, starting from the most prominent feature and adding other features later in the calculation. Logically speaking, it is a calculation that combines a lot of “if” statements. Since the goal of the research is to detect aggression, aggression score is chosen as the most important feature of the calculation. After that, additional scores are added in the equation to filter other possibilities.

   Different combinations have been tried with all three different datasets. Most notable combinations are aggression score combined with emoticon and emoji score as a possible neutralizer, aggression score combined with positive word score, emoji and emoticon as a

possible neutralizer, and aggression score with sentiscore, emoji, and emoticon as a neutralizer.

The most complete method uses these rules in sequence:

1. If no aggression presence and positive word score are more than 0, the sentence score will be positive.

2. If the aggression score is less than 0 (meaning there is aggression found), it will be added with positive word score. If the addition result is equal or less than 0 the sentence will be scored negative. This gives a chance for aggression to be neutralized by positive word presence.

3. If there is no aggression and positive word present and sentiscore are less than zero and less than sentiscore's mean threshold, sentence score will be negative.

4. If there is no aggression, no positive word presence, and no sentiscore calculated, the sentence score will be 0 or neutral.

5. Apply punctuation as an intensifier of sentence overall score.

If any sentence in a single input has a score less than 0, it will be added with emoji and emoticon score to check for possible neutralization assuming that emoji and emoticons presence indicates possible jokes between friends. If it is still less than 0, then the single input will be classified as aggression.

For example, the input "You are an idiot and a hater, you do know the difference between killing and irritating right?", will have a calculation of:

- Aggression score = -2.0, Sentiscore = -0.3, No positive word.

- Sentence score = -2.0 (because of rule number 1)

- Since there is only one sentence in the input and the sentence score is negative, then the single input will be marked as aggression.

The results of single type classification are sentence scores and a verdict of whether or not the single input is aggression. These results act as a final result for single type input and as a base result for further assessment in both QA and thread classification.

QA Classification

For QA type input, further processing using rules derived from experiments are needed as shown in Figure 3.5. The rules are as follows:

1. If the question section is marked as aggression, the whole QA will be marked as aggression.

2. If question section is not marked as aggression, check for answer section. If the answer is marked as aggression, go to step 3. If not, QA will be marked as non-aggressive.

3. If the answer section's aggression score cannot be neutralized by the answer section's positive word score, QA input will be marked aggression.

4. If not, check for laughter cue in the answer section, if it is present then QA will be marked as non-aggressive, otherwise, QA will be marked as aggression.

```
BEGIN classify QA

   if Q is aggression:
      QA is aggression
   else if A is aggression:
      if A's aggression_score + A's positive_score > 0:
         QA is aggression
      else if A has laughter:
         QA is not aggression
      else:
         QA is aggression

END
```

Figure 3.5 QA classification pseudocode

Thread Style Classification

Similar to QA classification, thread style classification (Figure 3.6) follows an independent set of rules defined as:

1. If half or more of entries in a thread are marked as aggression from single type classification stage, then the whole thread will be marked as aggression. In this step, only the verdict from earlier single type classification stage was considered.

2. If not, check for aggression score of each single entry in the thread.

3. A single entry will be marked as aggression if half or more of the sentences contain aggressive word. This aggression verdict of each single entry in the thread is independent from earlier verdict from single type classification stage.

4. If one or more single entry is marked as aggression then the whole thread will be marked as aggression. In this step, only the verdict from step (3) was considered.

```
BEGIN classify thread

  for each text_aggression_score:
    overall_aggression = 0
    cnt_aggression = 0
    for each aggression_score
      if aggression_score < 0:
        cnt_aggression = cnt_aggression + 1

    if cnt_aggression < length(aggression_score)/2:
      overall_aggression = 1

    push overall_aggression to array_overall_aggression

  if sum text_is_aggression > thread_entry_count/2:
    thread is aggression
  else if sum(array_overall_aggression) > 1:
    thread is aggression
  else:
    thread is not aggression

END
```

Figure 3.6 Thread classification pseudocode

3.2 Evaluation Methods

3.2.1 Dataset Validity Evaluation

There are three datasets used for this research: single style from Facebook, QA from Formspring.me, and thread style from MySpace.com. Since the datasets for QA and thread style were taken from past researches, there is no need to validate them anymore. However, for the Facebook dataset, which was built and annotated specifically for this research, an inter-annotator agreement coefficient using Cohen's Kappa value needs to be calculated.

Table 3.1 Annotation results

		Annotator 1	
		Y	N
Annotator 2	Y	210 (a)	23 (b)
	N	24 (c)	146 (d)

Table 4.1 shows the annotation results, where:

- a is the number of aggression agreed upon by both annotators
- b is the number of aggression marked by annotator 2 only
- c is the number of aggression marked by annotator 1 only
- and d is the number of non-aggressive input agreed upon by both annotators

From the annotation result specified in the table above, the process of calculating Cohen's Kappa is as follows:

Starting with calculating the agreement as P_o:

$$P_o = \frac{a+d}{a+b+c+d} = \frac{210+146}{210+23+24+146} = \frac{356}{403} = 0.88338$$

Then the probability of both annotators would label "yes" at random as:

$$P_{yes} = \frac{a+b}{a+b+c+d} \cdot \frac{a+c}{a+b+c+d} = \frac{210+23}{210+23+24+146} \cdot \frac{210+24}{210+23+24+146} = 0.57816 \cdot 0.58065 = 0.33571$$

And label "no" at random as:

$$P_{no} = \frac{c+d}{a+b+c+d} \cdot \frac{b+d}{a+b+c+d} = \frac{23+146}{210+23+24+146} \cdot \frac{24+146}{210+23+24+146} = 0.42184 \cdot 0.41936 = 0.17690$$

Then overall random agreement probability as:

$$P_e = P_{yes} + P_{no} = 0.33571 + 0.17690 = 0.51261$$

And finally, Cohen's kappa as:

$$\kappa = \frac{P_o - P_e}{1 - P_e} = 1 - \frac{1 - P_o}{1 - P_e} = 1 - \frac{1 - 0.88338}{1 - 0.51261} = 1 - \frac{0.11662}{0.48739} = 1 - 0.23927 = 0.76073$$

The kappa value shows that the dataset is deemed reliable, as 0.61-0.80 was presented as "substantial" or significant by Landis and Koch (1977) .

3.2.2 Performance Measure

Throughout the development period, the software was evaluated whenever significant changes are made. The evaluation was done using a specifically made software written in Python that calculates four different performance measurements: accuracy, precision, recall, and f measure. The predicted

result is compared to the actual result and from there the four performance measurements are calculated.

As discussed in Chapter 2 (Literature Review), accuracy alone cannot be relied on as a performance measure especially when the data is not symmetrical. Since the datasets being used are taken from different resources, it cannot be guaranteed that the data would be symmetrical. Also, in real life practice, it is also not possible that actual data would be symmetrical, hence accuracy percentage alone cannot represent a software performance. That is why the other three measurements are used: to give a more balanced approach to measuring the performance of the resulting algorithm.

For datasets taken from other studies, evaluation is done by comparing the performance measure of the current algorithm to past research. The performance measure for past research by Bayzick et al. (2018) and Reynolds et al. (2011) are briefly discussed in earlier section (2.2.1) and used during performance evaluation.

In Chapter 4 (Results and Discussions), results of both training and testing stage will be discussed. Further discussions of the software application on unlabelled data will also be presented.

Chapter 4

Results and Discussions

4.1 Finalizing elements on cyber aggression detection

After developing the algorithm and experimenting with features discussed in Chapter 3 (Methods), the latest version of the software was finalized (as shown in Figure 3.1 and specified in pseudocodes in Figure 3.2, 3.3, 3.4, 3.5, and 3.6) and chosen to be examined further to answer the question of cyber aggression elements in textual communication of online social media.

In the earliest version (0.0), the code's flow was linear, with a very simple sentiment analysis without negation and modifier handling, then if the sentiment is negative then aggressive words presence checking was done. This version gave many false negatives because the sentiment analysis was too straightforward and was not able to correctly classify an input.

In versions 1.x, weighted calculation was implemented instead of linear process. Although it had an improved performance, important features of sentiment analysis such as modifier and negation have yet to be implemented, meaning it had limitations on processing more complex sentences.

In versions 2.x, the weighted calculation was dropped and a rule-based approach really similar to the final version (3.8.3.9) was implemented. The main differences between version 2.x and 3.x are the application of more linguistical approach in the analysis process and reusability of the functions in the software. Full notes of tuning done throughout the development process can be found in Appendix A.

Throughout the development process, all three types of inputs were incorporated in the training. Since single type input was meant to be the foundation of other types, it was utilized during the beginning to the middle of the learning phase. Then functionality to process QA and thread was added by extending the functions from a single classification process. The learning graph of the algorithm can be found in Figure 4.1 and Figure 4.2 below. For complete performance log of training stage, refer to Appendix A.

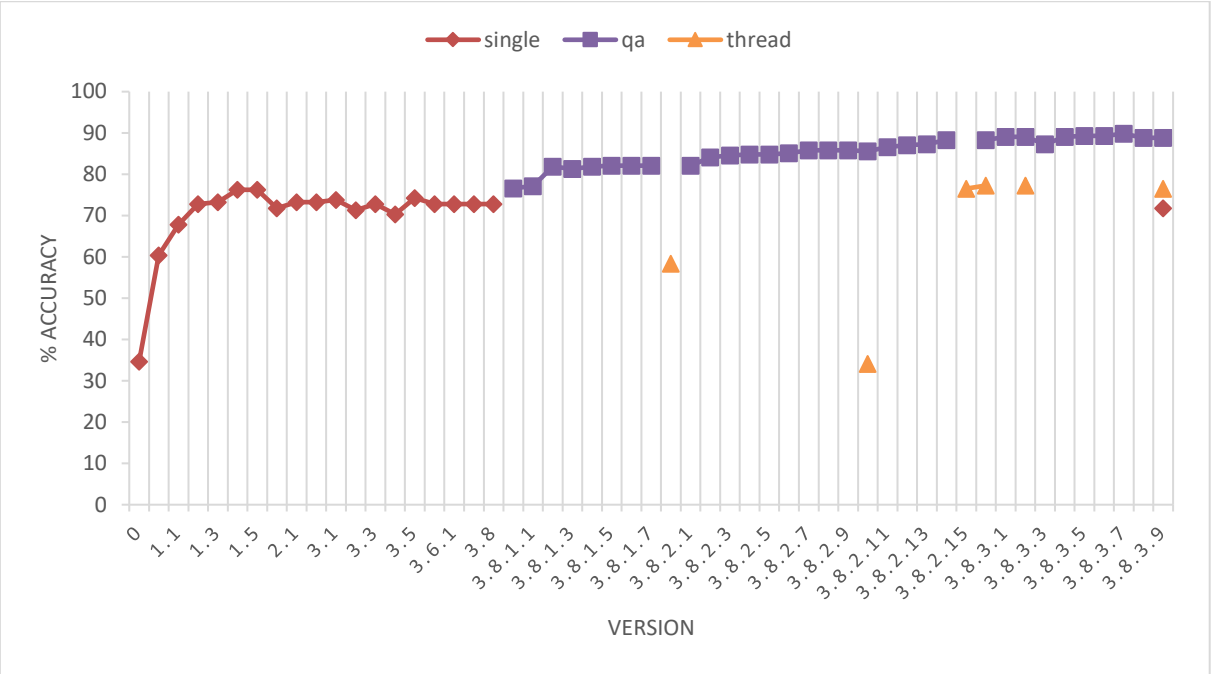

Figure 4.1 Algorithm learning graph (accuracy)

As seen in Figure 4.1, in the final version of 3.8.3.9, QA type showed the highest accuracy when compared to single and thread type. This was expected due to the extensive development process from version 3.8.x that has been focused on QA type as training dataset and as stated in Chapter 3, the dataset for QA is balanced between aggression and non-aggressive entries. During the development process focusing on QA type, several adjustments on single type processing were also done, resulting in an inconsequential decline in accuracy for single type classification.

Also, as discussed in Chapter 2 (Literature Review), accuracy alone cannot be justified as a sole performance measure in classification task because it does not provide a balanced insight for the unbalanced dataset. Thus in this research, accuracy is not established as the main indicator of the algorithm's performance. This leads to another learning curve showing the growth of F1 score, which is selected as the main performance review indicator.

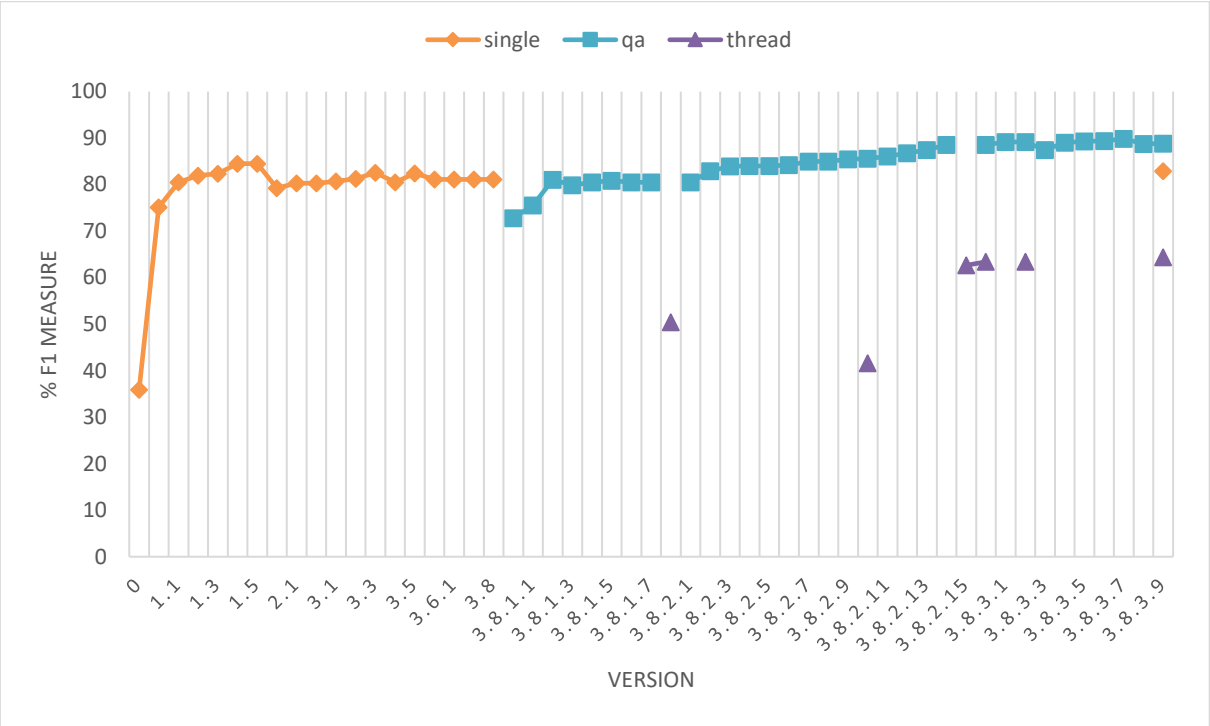

Figure 4.2 Algorithm learning graph (F1 Measure)

In Figure 4.2, it can be seen that the although the F1 measure of QA and single type input is more than 80%, the F1 for thread type is quite low with only 64.368%. This seemingly low value was accepted because it is really close to the original research's performance which is 64.407%. More details regarding the comparison of this research to each dataset's original research performances can be found in the later section (4.2.3). Also, unlike the accuracy measure, where the last version's single type accuracy was recorded lower than version 3.8, the F1 score for single type processing actually rose from 81.099% to 82.883%.

Since QA and thread processing are extensions of single type input processing, changing rules for QA or thread type processing would not affect the performance of the remaining two types. It can be seen from the graph that even though not every input type was tested against each version, in the last version (3.6.3.9) the performance remained stable for all three types. Of course, this also means that changing rules of single type classification might affect the performances of QA and thread classification. This could be an issue if the scores produced by Text Processing stage were not carried on to QA and thread classification stage.

However, since all scores (aggression score, positive word score, sentiscore, emoji and emoticon scores) are also shared with QA and thread classification function in the algorithm, QA and thread classification can operate independently without relying on single type classification verdict. This means, all three classification functions (single type, QA, and thread) are clearly separated and can

easily be modified individually in case of future development. For this research, the classification rules are as discussed in Chapter 3 section 3.1.2.

After the development process was completed, the last version (3.8.3.9) of the algorithm with the most stable performance across datasets was then run to answer the research question regarding elements of cyber aggression detection in different types of social media texts. This was done by creating several alternate sub-versions, where the only certain score is included in the classification. The details of the comparison of sub-versions can be seen in Table 4.1 below. The values in bold are the highest scores for each type in each performance measure.

Table 4.1 Version 3.8.3.9 elements performance against training datasets comparison

Elements	Dataset	Performance measure				%F1 Mean Value
		%Accuracy	%Precision	%Recall	%F1 Measure	
All	Single	71.782	71.875	97.872	82.883	78.695
	QA	88.806	88.614	89.055	88.834	
	Thread	**76.515**	**50.000**	**90.323**	**64.368**	
Aggression score	Single	71.287	71.066	**99.291**	82.840	75.459
	QA	**90.049**	**93.048**	86.567	**89.691**	
	Thread	72.727	44.681	67.742	53.846	
Aggression score, emoji, and emoticon	Single	71.287	71.066	**99.291**	82.840	75.459
	QA	**90.049**	**93.048**	86.567	**89.691**	
	Thread	72.727	44.681	67.742	53.846	
Aggression score, emoji, emoticon, and positive word score	Single	**73.762**	**73.404**	97.872	**83.891**	75.423
	QA	88.806	90.625	86.567	88.549	
	Thread	72.727	44.681	67.742	53.846	
Aggression score, emoji, emoticon, and sentiscore	Single	70.297	70.352	**99.291**	82.353	78.537
	QA	88.806	88.235	**89.552**	88.889	
	Thread	**76.515**	**50.000**	**90.323**	**64.368**	

There are four elements to be assessed, aggression score, positive word score, sentiscore, and emoticon-emoji score. The goal of the comparison is to find which features actually contribute the most to algorithm performance and finally decide which sub-version can be expected to give the most stable result across data types. More details on each performance measure comparison and significance are as follows:

- Accuracy

Accuracy, being the most intuitive performance measure, provides information regarding how well the algorithm can correctly classify inputs. Earlier in Chapter 2 (Literature Review), it was discussed how accuracy cannot be relied on to measure performance on the unbalanced dataset. The statement is consistent with the data presented in Table 4.1, in which the difference between the highest and lowest accuracy percentage for unbalanced (single type and thread type) dataset are 3.47 and 3.79 as opposed to the balanced (QA) dataset's 1.24. Following this knowledge, accuracy will only be used as a reference to overall performance instead of the main performance measure.

It can be seen from Table 4.1 that for single post, a combination of aggression score, emoji and emoticon score, and positive word score performed best with an accuracy of 73.762%. The lowest accuracy came from a combination of aggression score, emoji and emoticon score, and sentiscore with 70.297%. This suggests that positive word presence plays a more important part in determining aggression compared to sentiscore.

For the thread dataset, the combination of all elements gave the best accuracy with 76.515%, equal to the combination of aggression score, emoji and emoticon score, and sentiscore. In all other three subversions, the accuracy is 72.727%. It shows that utilizing positive word score as possible neutralizer actually gave a negative impact on the algorithm's performance.

While for QA type, the algorithm utilizing only aggression score performed the best with over 90%, interestingly, combining aggression score with emoticon and emoji score gave the exact same result, meaning that emoticon and emoji have little to no impact to performance. All three remaining subversions have an equal accuracy of 88.806%. This indicates that positive word presence contributes to better accuracy compared with sentiscore.

- Precision

The next performance measure to be reviewed is precision, which gives information regarding the algorithm's performance in avoiding false positives. This means that precision shows how many entries are correctly labelled as aggressive, meaning the value will get lower if the algorithm is set to be too sensitive to aggression detection.

Higher precision is expected to correspond with the algorithm's ability to find neutralizing elements to aggression such as emoji and emoticon and positive words presence. This corresponds to the data presented in the table, where it can be seen that for single type dataset, the highest precision came from sub-version combining scores from aggression,

emoji and emoticon, and positive words with 73.404%. The lowest precision score of 70.352% is from a subversion with all scores but positive word score, which, again, corresponds to the expectation that positive word presence acts as a possible neutralizer.

In thread dataset, the best precision score came from a combination of all elements and a sub-version combining aggression, emoji and emoticon, and sentiscore with 50%. In both dataset types, emoji and emoticon score and positive word score acts as a possible neutralizer to aggression score. Similar with accuracy score, all precision scores of the remaining three subversions are equal to 44.681%.

Interestingly for QA dataset, the highest precision actually came from sub-version with aggression score only with the score of 93.048%. The lowest precision of 88.235% can be seen on sub-version combining all scores except for positive word score. This is an unexpected result, unlike the precision of single type and thread type discussed earlier because the precision for sub-version utilizing possible neutralizer actually fell short compared to the one without neutralizer. This could be attributed to the fact that sentiscore may act as both possible neutralizer and amplifier in aggression detection, and in this case, the lack of sentiscore in the equation actually made up for less false positive counts.

- Recall

The next performance measure is used to evaluate how well the algorithm can classify input with aggression or how many of the actual aggression can the algorithm detect. Of course, in a matter of identifying cyber aggression, some might argue that recall is the most important performance measure. It is expected that higher recall corresponds to a stricter rule of detecting aggression, meaning that possible neutralizer might actually lower recall score.

The assumption is proven to be correct according to data in Table 4.1, where sub-version utilizing positive word score gave the lowest recall on all three datasets with a score of 97.872% for single type input, 86.567% for QA input, and 67.742% for thread. Another interesting finding is that the sub-version combining scores from aggression, emoji and emoticon, and sentiscore gave the best recall for every dataset. This is because sentiscore can be chosen as aggression indicator, as detailed in single type classification section in Chapter 3. Although other sub-versions that did not include positive word score also performed relatively well across datatypes, only the one with sentiscore actually showed consistency for all datatypes.

For single type dataset, the highest recall actually reached more than 99%, meaning the algorithm can identify aggression almost all the time. When using a combination of all scores, however, the recall is recorded lower, which is expected because when possible neutralizer is implemented, there will be some entries that are no longer detected as aggression because of the presence of neutralizer such as positive word.

As opposed to single type's reliance on aggressive word score, the performance of sub-version relying solely on aggression score was actually disappointing for thread type dataset. Somehow, the sub-version with complete elements actually worked best with the thread dataset. This is due to the fact that individual entry in a thread usually has longer characters compared to single type data, the longer data means more significant and accurate sentiscore, which can make a negative input more negative (as explained in the classification section from Chapter 3).

- F1 Measure

In some cases, precision might get really low when the main concern is maximizing recall, therefore a more balanced approach is needed. The F1 measure gives a more balanced approach towards recall and precision. By assessing the F1 measure, the overly conscious approach can be avoided because maximizing the algorithm's ability to correctly identify aggression is the main concern of this research. Correctly identifying aggression means that the algorithm is expected to be able to balance between finding aggression and non-aggression instead of blindly labelling inputs as aggression without checking for a possible neutralizer.

The F1 score for the thread dataset can easily be seen as the lowest compared to other data types, with the highest F1 of only 64.368%. This is expected due to the low score of precision as discussed earlier in previous section

In order to find the most stable sub-version, the mean value of each sub-version F1 score was calculated. It can be seen that the sub-version with complete elements is the one with the highest F1 score mean of 78.695% compared to the lowest of 75.423% for sub-version combining all scores but positive word score. This means combining all elements (scores) is the most stable method across dataset types. Therefore the sub-version utilizing all elements is chosen as the final algorithm would be tested using testing datasets for further performance check.

Table 4.2 Comparison of current final algorithm performance to QA and thread type original research performance (Bayzick et al., 2018; Reynolds et al., 2011)

Performance Measure	QA type	Original QA dataset (Reynolds et al., 2011)	Thread type	Original thread dataset (Bayzick et al., 2018)
Accuracy	88.806	Not specified	76.515	83.969
Precision	**88.614**	30.600	50.000	51.351
Recall	**89.055**	40.500	**90.323**	86.364
F1 Score	**88.834**	34.861	64.368	64.407

For QA and thread dataset, a comparison to the dataset's original research performance record is also needed. QA dataset was used in research by Reynolds et al. (2011), where it was claimed that the resulting software managed to reach recall of 96.6% with the cost of very low precision score. Their most balanced approach, however, only managed to reach 40.5% recall. Compared to Reynolds et al.'s research result, the final algorithm provides better performance with an F1 score of 88.834%. The details of the comparison between this research with Reynolds et al.'s "best trade-off" performance can be seen in Table 4.2 above.

For thread dataset, the algorithm's 64.368% F1 score is only lower than Bayzick et al. (2018) F1 score by 0.039%. But the algorithm managed to outperform Bayzick et al.'s software in the recall, with 90% compared to 86%. This means the final algorithm of this research managed to detect more aggression compared to the dataset's original research software.

4.2 Final algorithm performance with testing datasets

This section provides information on how the final algorithm version performed against testing datasets from all three input types. In order to prove the reliability of the testing stage results, three datasets were used for each data type. From 356 annotated single type texts it was divided into datasets of 100s and 156 texts. For QA type, 1000 balanced entries were divided into groups of 300s and 400. Thread style datasets were originally grouped into packets so there was no need to separate them anymore.

4.2.1 Single type testing

Table 4.3 Algorithm performance with single type datasets

Data set	TP	TN	FP	FN	Total Actual Pos	Total Actual Neg	Accuracy	Precision	Recall	F1 Score
1	53	28	18	1	54	46	81.000	74.648	98.148	84.800
2	58	25	17	0	58	42	83.000	77.333	100.000	87.218
3	88	44	14	10	98	58	84.615	86.275	89.796	88.000
						Average	82.872	79.419	95.981	86.673

The final algorithm performed well on all three single type datasets and even achieved 100% recall for the second dataset, as seen in Table 4.3. Remarkably, it actually showed an even better performance compared to when it was used on the training dataset. The lowest accuracy achieved is 81%, higher than even the highest performance of the sub-version discussed before. The same also happened for precision and F1 measure, where the lowest score in processing testing datasets are higher than the best score from the training dataset.

However for recall, even though it reached 100 for the second dataset, the average of recall for all three datasets is only 95.98%, lower than even the lowest recall from the training dataset. This means that the overall performance got higher while the algorithm's ability to actually recognize aggression input got slightly lower. All in all, it can still be considered a stable performance from training dataset to testing datasets.

4.2.2 QA type testing

Table 4.4 Algorithm performance with QA type datasets

Data set	TP	TN	FP	FN	Total Actual Pos	Total Actual Neg	Accuracy	Precision	Recall	F1 Score
1	28	232	39	1	29	271	86.667	41.791	96.552	58.333
2	113	126	49	12	125	175	79.667	69.753	90.400	78.746
3	284	40	14	62	346	54	81.000	95.302	82.081	88.199
						Average	82.444	68.949	89.678	75.093

The algorithm's performance on QA testing dataset shows more variety in precision and subsequently F1 measure. The whole testing dataset consists of 1000 balanced entries, but when it was divided into three different datasets it became three differently balanced datasets. Dataset 1

only has less than 10% aggression, while dataset 2 is almost balanced with a ratio of 41:59 of aggression and non-aggression entries and dataset 3 has 86.5% aggression.

Table 4.4 shows that for dataset 1, the algorithm failed to correctly identify false positives, leading to a low precision of 41.791% and an F1 score of 58.333%. It performed better on the almost balanced dataset 2, with an F1 score of 78.746%. Surprisingly, the algorithm shows the best performance when it was used to classify dataset with heavy aggression entries.

This means the extended rules for QA classification has managed to capture the essence of textual aggression, given the satisfactory recall on all three datasets. But unfortunately, it is still not refined enough to identify non-aggression texts, resulting in low overall performance when faced with datasets with a heavy number of non-aggressive texts.

Unlike the single type data classification, unfortunately, the algorithm overall performance actually dropped compared to training data performance measure. This could be caused by two factors, the first one is overfitting, given the long and strenuous period of development specifically for the QA training dataset. The second possible factor is the fact that the training dataset was intentionally balanced between the number of non-aggressive and aggressive entries.

Lastly, it is important to compare the testing dataset performance results to the dataset's original software performance results. As discussed before in chapter 4.1, Reynolds et al. (2011) recorded the highest recall of 96.6% with precision less than 10%. With QA dataset 1 (which is the dataset with lowest overall performance), the algorithm managed to reach similar recall while maintaining the precision of 41.791%.

4.2.3 Thread type testing

Table 4.5 Algorithm performance with thread datasets

Data set	True Pos	True Neg	False Pos	False Neg	Total Actual Pos	Total Actual Neg	Accuracy	Precision	Recall	F1 Score
1	27	37	62	6	33	99	48.485	30.337	81.818	44.262
2	72	28	113	13	85	141	44.248	38.919	84.706	53.333
3	0	97	0	108	108	97	47.317	0.000	0.000	0.000
						Average	46.683	34.628	83.262	48.798

Table 4.5 shows that for thread type dataset, the performance dropped quite significantly compared to its performance on the training dataset. The accuracy dropped from 76.515% to at most 48.485%,

and the precision dropped from 50% to less than 40%. Of course, this also means that the F1 score also dropped from almost 65% to less than 55%. Recall measure also dropped, though it still managed to reach more than 80%.

From the data, it can be concluded that the thread classification rules are enough to identify aggressive inputs but it is certainly lacking in discerning non-aggressive inputs. Comparing the performance to the report on Bayzick et al. (2018) research, the original research using the datasets shows that the algorithm performs better on dataset one, where 37 non-aggressive inputs are identified by the algorithm as opposed to Bayzick's record of 29. In terms of accuracy, the algorithm managed to outperform the original study with 48.485% compared to 41.216%. Unfortunately, the algorithm fell short compared to Bayzick's software performance on dataset 2 and 3, where it managed to reach an accuracy of 58.407% and 65.7%.

Next section will provide more information and discussion regarding the results of the algorithm's classification of two different unlabelled single type datasets.

4.3 Application with unlabelled data

After testing the final algorithm with testing datasets, the last step is to utilize the algorithm to classify unlabelled data. The unlabelled data chosen is public comments from Facebook page, meaning the data type is single type. There are several reasons as to why single type data is chosen as verification dataset:

1. Single type classification is the root of all classification functions, so it needs to be explored more.

2. Getting single type data is easier compared to QA and thread data.

3. Based on the performance evaluation during training and testing stage, the algorithm shows the best performance when it is used to classify single type dataset.

The sources chosen are comments from Lorde and Donald Trump's Official Facebook page. Both are famous public figures from English speaking countries so it is expected that the data taken from their pages are mostly in English. The difference between the two figures is mainly public opinion about them. Lorde, as a singer, does not really have many controversies compared to Donald Trump, who is a famous businessman, television personality, and the current president of the USA. It is also a common belief that many people despise Donald Trump, hence more aggression is expected from his official page. It is worth noting that Facebook, as one of the most widely-used social media

nowadays, has been considered to have a high level of aggression. This is backed up by the finding that cyber aggression presence corresponds to the most used social media, as stated by Whittaker and Kowalski (2014).

There are 1000 comments randomly taken from Lorde and Donald Trump page and the algorithm was run separately on the datasets. The data was scraped from Facebook in April 2018 and mostly collected from the period of Trump's campaign and his early presidency. To comply with this research's single type input training and test dataset, only comments with a character count of equal to or less than 150 are selected. The results are presented in Table 4.6 below. Note that the accuracy of single style data classification has been proven to be more than 80%.

Table 4.6 Unlabelled data result comparison

Facebook Page	Aggression	Non-aggressive	% of Aggression
Lorde	265	735	26.500
Donald Trump	350	650	35.000

In accordance with the assumption that Donald Trump is a less likeable figure compared to Lorde, out of 1000 comments in his official Facebook page, 350 are classified as aggression. The aggression percentage is not as high as expected, probably due to the fact that the data collected was mostly taken from Trump's period of campaign and early presidency, when a lot of his supporters tried to show their supports through Trump's page. As surprising as it seems though, even Lorde, a singer without known controversies, has been a target of cyber aggression, with 26.5% of analysed aggression no less. It could be concluded that cyber aggression can happen to everyone, whether it is someone that is supposedly "likeable" or "hated".

Chapter 5

Conclusion

In this chapter, summary of this research's outcome is discussed, followed by discussion of the limitations, and finally the insight on possible future work.

5.1 Research outcome

With the popularity of online social media, cyber aggression has been a prevalent issue in human cyber communication. Cyber aggression, while seemingly harmless, actually has many bad effects just like traditional bullying, including depression, academic issues, and even suicide attempts (Bauman et al., 2013; Hinduja & Patchin, 2008; Pettalia et al., 2013). Thus, exploring how to detect textual cyber aggression is very important.

In an attempt to accomplish the research objectives and answer the research questions detailed in Chapter 1 (Introduction), this research has studied past works on cyberbullying and harassment and came with a conclusion that the number of aggression happen in social media corresponds to the popularity of the media itself. In this regards, Facebook has been chosen as one of the sources of the dataset. The other two datasets were taken from past studies, where it was claimed that the media was known to have a high level of aggression.

Three different types of input (single type, QA, and thread) were used as datasets to show that the algorithm can work across platform, a limitation that has been pointed out in research by Ruths and Pfeffer (2014) paper on using social media in behaviour study.

Another conclusion drawn from studying past work is regarding the most suitable technique to detect cyber aggression in textual input. Although machine learning has been popular in recent years, many studies have shown that rule-based approach using lexicon as enhancement is highly accurate and relatively easy to implement compared to machine learning because of lack of high volume of labelled data that is needed to train machine learning techniques.

After choosing and exploring further on lexicon enhanced rule-based approach, an algorithm to detect textual aggression was developed, tuned, and finalized. The final algorithm uses combination of scores or features such as text aggression score, text sentiment score, positive word score, emoji score, and emoticon score to determine whether an input is classified as aggression or not. The most important feature of the resulting algorithm is its ability to process three different types of inputs

and the high reusability degree of the functions in the algorithm, making it easy to extend the code to include more types in the future or modify current rules of each type classification.

It was also found that emoji, which is developed and released by an authorized organization, is not likely to intentionally include aggression elements. Emoticons, while not as monitored as emoji, are also less likely to pose aggression in texts. Thus, both emoji and emoticons are considered as a possible neutralizer to aggression in classifying textual aggression.

The algorithm was then validated using performance measure of accuracy, precision, recall, and F1 measure, with the last one considered to be the most important measure because it gives a balanced approach to both true positives and false positives presence. Based on the testing datasets, the algorithm shows promising performance for both single type and QA type inputs, with an average F1 score of 86.673% and 75.093% correspondingly. With these F1 scores, it can be said that the rules defined for single type and QA type classification are able to distinguish aggressive and non-aggressive inputs effectively.

Unfortunately, the average F1 score for thread style is not as satisfying with only 48.798%. This low number is due to the low precision score (34.628% on average) even though it has a satisfying recall of 83.262%. This indicates that the rules defined for thread style classification are sufficient to recognize aggression but inadequate in recognizing non-aggressive post, falsely marking many non-aggressive inputs as aggression.

In conclusion, the resulting algorithm of this study managed to outperform previous study on aggression detection for QA input type while also being able to process different types of text including single type and thread. The reusability of the algorithm also serves as an easy foundation for future developments.

5.2 Limitations

As mentioned in section 5.1, this research's limitation lies in the low precision in classifying thread style inputs. Even for QA and single type input, the precision is not as high as the recall measures, meaning that while the algorithm may have grasped the essence of detecting aggression, its ability to detect non-aggressive inputs still needs to be explored further. Though as discussed before in Chapter 2 (Literature Review), precision is a measure that is more important in research where false positive cannot be tolerated. In detecting aggression though, some might argue that recall is a more important measure compared to precision because it is deemed better to label a text as "possible aggression" rather than falsely labelling aggression as non-aggressive text.

But since this research has put a lot of emphasis in the F1 score, which is a score derived from balancing both precision and recall, it means that the algorithm has yet to achieve a balanced performance for all data types especially thread style. Of course, the method used in this research's algorithm has many limitations, which is discussed further in section 5.3 as future work enhancements.

The lack of standard definition of what cyberbullying also brings forth another limitation. Since all datasets used in this research were compiled by separate research, this means each research may have its own definition of cyberbullying and how to recognize it in a textual context. This issue also affects how each annotator perceive cyberbullying detection because even though the annotators for single type dataset were given proper guideline about how to correctly identify a cyberbullying text, others might not have done the same. Therefore, it is important for a standardized definition of cyberbullying to be released by authority, to avoid ambiguity in future research.

5.3 Future work

To improve the algorithm's limitations, there are some enhancements that have been studied briefly but cannot be explored further due to this research's time limit. In the future, several suggestions on which features to be added or modified can be found as follows:

- Exploring the connection between sentences

  Sentences in the same input are usually connected to each other, and this has yet to be explored in this research. For example, an input of "You are pretty. Lies!". As it can be seen, there is a definitive connection where the second sentence actually cancels the first sentence.

- Direct translation of emoticon to representative sentence

  Although it was stated before that emoticon is considered as possible aggression neutralizer in this research, it may prove to be essential to actually translate emoticon to its representative sentence or word such as :-) to "smiling face". This is not applicable for current research because unlike emoji, emoticons are not regulated by an organization, meaning there is no standard on how to translate an emoticon. It also has to be noted though, that in years to come, emoticons might not be used anymore as it will be replaced by emoji.

- Exploring group of words as a phrase

There are many aggressive phrases that cannot be detected using this research's finalized algorithm because it explores tokens as individual words, not as a phrase. Examples of aggressive phrases are "son of a b*tch" and "suck my d*ck".

Another reason to explore phrases instead of an individual word is that the algorithm might be able to distinguish if an aggressive word is directed to someone or simply used neutrally. For example, the word "death" is listed as an aggressive word, but when it is used in a neutral sentence such as "Do you believe in life after death?", it is not an indicator of aggression.

- An aggression dictionary with more details

  Existing dictionaries of aggressive or offensive word only list the words or phrases itself, without further information unlike SentiWordNet, which includes much information such as its POS tag, synonym, example sentence, and many more. In order to correctly approach textual aggression detection in a more linguistical manner, a comprehensive aggression dictionary is needed.

- Elongated words as an intensifier

  In this research, elongated words are simply shortened but the information of it being elongated was not stored nor explored further. Instinctively, elongated words are supposed to give a stronger feeling, for example, compare "ugly" to "uglyyyyy".

Appendix A

Training Stage Version Notes and Performance Log

This appendix shows the complete notes of development (training) process, starting with a table of the algorithm versions notes and change logs during the development process and the performance measure score of each version. Then a table of each version's performance using training data is presented. Last is a table of testing stage performance.

A.1 Algorithm version change logs

The table below describes the algorithm's version during the development process. The underlined input style(s) marked which was used for performance measurement.

Table A.1 Algorithm version notes

Version	Input style supported	Notes
0.0	• Single style	• Initial prototype • Swearwords used are taken from noswearing.com only, consisted of 349 words • Emoji was separated from text and calculated for emoji sentiment using emoji sentiment formula shown in Chapter 4 • Text was processed using a linear method: very simple sentiment analysis using bag-of-words method, then texts with negative sentiment score are checked for presence of swearwords • Sentiment analysis was done using SentiWordNet • Capitalization was applied as intensifier that will affect the whole input's sentiment score • Excessive punctuation count was used as intensifier
1.0	• Single style	• Swearwords list expanded using words from bannedwordlist and cs.cmu.edu, consisted of 1619 unique words • Aggression detection relied solely on presence of swear words
1.1	• Single style	• Fixed a bug where input without sentiment score was not processed further • Aggression was assessed using scoring system where sentiment score weighed 30% and aggressive words presence weighed 70% • Scoring system was only triggered when one or more aggressive words is found in input

		• Input would be marked as aggression when overall score is less than 0
1.2	• Single style	• Word “hell” was removed from swearword list for experimental purpose because it has been found in many non-aggressive inputs.
1.3	• Single style	• Score intensifying process from capitalization was changed to affect only corresponding word(s), for example, “You are so UGLY and stupid” only the word ugly would be intensified
1.4	• Single style	• Fixed capitalization percentage intensifying rate wrong calculation process • Removed the word “god” from swearword list because of its ambiguous nature
1.5	• Single style	• Text was broken down into sentences and processed separately, then the score was aggregated as overall score • Input would be marked as aggression if any sentence contains aggressive words, making overall score as a scale of severity rather than aggression determinant
2.0	• Single style	• Weighted scoring system was changed to rule-based approach, where occurrence of aggressive words would give a negative score (the score was just a static scalar) • For multi sentences input, if more than half of the sentences contain aggression then the input would be marked as aggression
2.1	• Single style	• Fixed bug on aggression count • Removed the word “kid” from swearword list
3.0	• Single style	• Score applied for occurrence of swearword was determined using relative frequency function • Tidied up the algorithm for “text clean up” function reusability
3.1	• Single style	• Stop words removal was applied • Lemmatization was dropped
3.2	• Single style	• Began to separate text scores into two different scores: sentiment score and aggression score • Weighted scoring system was reapplied and if final score was less than 0 then it would be marked as aggression • Added simple modifier word handler during text sentiment analysis, modifier word search was only done for three steps backwards

3.3	• Single style	• Added negation handler during text sentiment score calculation • Functions of modifier handler, negation handler, and sentiment score calculation were separated for reusability • When checking for word's sentiment score in SentiWordNet, stop words and modifier words were excluded
3.4	• Single style	• "Apply modifier" functions were applied to aggression score analysis too
3.5	• Single style	• Rule-based approach was reapplied instead of weighted scoring system • The rule-based approach was the fundamental of final version's "single style" classification
3.6	• Single style	• Added function of "least check" • Added function of "but check" • Added handler of "n't" in negation handling
3.7	• Single style	• When getting a word's sentiment score from SentiWordNet, every synset would be looped and the most negative one would be chosen. Before this, the first found synset was chosen
3.8	• Single style	• Fixed error when only emoticon presents • Started to use QA style input, but was still treated as single style input
3.8.1.1	• Single style • QA style	• Added initial parameter to differentiate between single style to QA style input • When QA style was processed, it followed the same process with single style but with added QA classification • QA classification was done simply as if any section considered aggression then the whole QA would be marked as aggression
3.8.1.2	• Single style • QA style	• Removed some found ambiguous swearwords such as color, colored, babe • "weighted" system was reinvented, but instead of percentage, overly negative sentence would append the score list twice. For example, if before the score list would be [-1, 0.4, 0.5, 0.1], since -1 is supposed to be very negative it was changed to [-1, -1, 0.4, 0.5, 0.1], so the overly negative sentence would have more influence on classification process
3.8.1.3	• Single style • QA style	• Added function to check for pronoun whenever aggressive word was found

		• Swearwords with POS tag of noun or adverb were not checked for pronoun
3.8.1.4	• Single style • QA style	• Elongated words were shortened using shortening algorithm
3.8.1.5	• Single style • QA style	• Added pronouns “shes” and “hes” to pronoun list • Added function to normalize NIV words, extending previous version’s shortening algorithm with PyEnchant similar words suggestion function
3.8.1.6	• Single style • QA style	• Emoticons not separated by space were handled properly • Static threshold of sentiscore to be considered as aggression was changed to -0.1
3.8.2	• Single style • QA style • Thread style	• All functions were separated properly and have the same structure with final version • Added thread style classification function for thread input • Thread classification was simple rule of if any single input in a thread was marked as aggression during single style classification then the whole thread would be marked as aggression
3.8.2.1	• Single style • QA style • Thread style	• Rule-based single style classification was restructured for readability
3.8.2.2	• Single style • QA style • Thread style	• Laughter text processing added • Fixed wrong pronoun checking of aggressive word detection

3.8.2.3	• Single style
	• QA style
	• Thread style
		• To get sentiscore overly negative benchmark, the average of sentences sentiscore was calculated
		• QA classification rules were defined to finalized version of:
		  ○ If Q section is marked as aggression then the whole QA is aggression
		  ○ If Q section is not marked as aggression then QA will be marked as aggression only if A section has aggression score

3.8.2.4	• Single style • QA style • Thread style	• Laughter cue was used as possible neutralizer in A section
3.8.2.5	• Single style • QA style • Thread style	• Added pronoun neutralizer function
3.8.2.6	• Single style • QA style	• Code clean up

	• Thread style	
3.8.2.7	• Single style • QA style • Thread style	• Applied FuzzyWuzzy to calculate distance when determining best replacement word during NIV normalization
3.8.2.8	• Single style • QA style • Thread style	• Separated laughter checking function for reusability and readability
3.8.2.9	• Single style • QA style • Thread style	• Slang resolution was moved to before tokenizing process • Stop words removed was reactivated
3.8.2.10	• Single style • QA style • Thread style	• Pronoun search during aggression word detection was changed to be no longer using static backwards steps but flexible, until the first found noun or preposition before or after an aggressive word
3.8.2.11	• Single style • QA style • Thread style	• Fixed pronoun resolution to differentiate between “my [noun]” to “your [noun]”
3.8.2.12	• Single style • QA style • Thread style	• If sentiment score is negative, it will make sentence score more negative
3.8.2.13	• Single style • QA style • Thread style	• Single style classification was changed to “if any sentence in a single input has negative sentence score then the whole input is marked as aggression”
3.8.2.14	• Single style • QA style • Thread style	• Added swearwords list with list from thesis by Engman (2016) and removed every ambiguous word. This is the final swearword list.
3.8.2.15	• Single style • QA style • Thread style	• Changed rules for thread classification, where if 2 or more consecutive single inputs in a thread has aggressive words detected then the whole thread would be marked as aggression
3.8.3	• Single style • QA style • Thread style	• Added positive word checking
3.8.3.1	• Single style • QA style • Thread style	• Applied positive word presence as neutralizer in QA classification
3.8.3.2	• Single style • QA style • Thread style	• Code tidied up
3.8.3.3	• Single style • QA style • Thread style	• Added function to get count of aggressive word for comparison with positive word count

3.8.3.4	• Single style • QA style • Thread style	• Scoring system for aggressive word was simplified to fit positive word calculation • Removed function added in version 3.8.3.3 • Rules for QA classification was finalized as described in chapter 4
3.8.3.5	• Single style • QA style • Thread style	• Added pronoun normalization entry
3.8.3.6	• Single style • QA style • Thread style	• Emoji and emoticon scores were calculated as sum of each emoji and emoticon, not average
3.8.3.7	• Single style • QA style • Thread style	• Backwards search for modifier words would stop when reaching boundary
3.8.3.8	• Single style • QA style • Thread style	• Negation check backwards search was still limited to three steps but will stop when encountering boundary • Stop words removal was moved to after POS tagging process
3.8.3.9	• Single style • QA style • Thread style	• Every modifier constant was taken from VADER • This is the final version

A.2 Training stage performance measure

Below is the table detailing performance measurements during development process. Most versions were trained using one dataset only, then minor adjustments were made to cater to different dataset.

Table A.2 Training stage performance measure

Version	Dataset	Performance Measure			
		Accuracy	Precision	Recall	F1 Measure
0.0	Single training	34.65347	56.92308	26.24113	35.92233
1.0	Single training	60.39604	66.85083	85.8156	75.15528
1.1	Single training	67.82178	69.79167	95.03546	80.48048
1.2	Single training	72.77228	76.21951	88.65248	81.967213
1.3	Single training	73.26733	76.36364	89.3617	82.352941
1.4	Single training	76.23762	77.51479	92.9078	84.516129
1.5	Single training	76.23762	77.51479	92.9078	84.516129
2.0	Single training	71.78218	81.34328	77.30496	79.272727
2.1	Single training	73.26733	82.70677	78.01418	80.291971
3.0	Single training	73.26733	82.70677	78.01418	80.291971
3.1	Single training	73.76238	82.83582	78.7234	80.727273
3.2	Single training	71.28713	74.55621	89.3617	81.290323

3.3	Single training	72.77228	74.71264	92.19858	82.539683
3.4	Single training	70.29703	74.2515	87.94326	80.519481
3.5	Single training	74.25743	78.70968	86.52482	82.432432
3.6	Single training	72.77228	78.66667	83.68794	81.099656
3.7	Single training	72.77228	78.66667	83.68794	81.099656
3.8	Single training	72.77228	78.66667	83.68794	81.099656
3.8.1	QA training	76.61692	86.89655	62.68657	72.83237
3.8.1.1	QA training	77.11443	81.14286	70.64677	75.531915
3.8.1.2	QA training	81.8408	84.78261	77.61194	81.038961
3.8.1.3	QA training	81.34328	86.62791	74.12935	79.892761
3.8.1.4	QA training	81.8408	86.78161	75.12438	80.533333
3.8.1.5	QA training	82.08955	86.85714	75.62189	80.851064
3.8.1.6	QA training	82.08955	88.16568	74.12935	80.540541
3.8.1.7	QA training	82.08955	88.16568	74.12935	80.540541
3.8.2	Thread training	58.33333	35	90.32258	50.45045
3.8.2.1	QA training	82.08955	88.16568	74.12935	80.540541
3.8.2.2	QA training	84.0796	89.59538	77.11443	82.887701
3.8.2.3	QA training	84.57711	87.56757	80.59701	83.937824
3.8.2.4	QA training	84.82587	88.88889	79.60199	83.989501
3.8.2.5	QA training	84.82587	88.88889	79.60199	83.989501
3.8.2.6	QA training	85.07463	89.38547	79.60199	84.210526
3.8.2.7	QA training	85.8209	90.44944	80.0995	84.960422
3.8.2.8	QA training	85.8209	90.44944	80.0995	84.960422
3.8.2.9	QA training	85.8209	87.5	83.58209	85.496183
3.8.2.10	QA training	85.57214	85.57214	85.57214	85.572139
	Thread training	34.09091	26.27119	100	41.610738
3.8.2.11	QA training	86.56716	89.30481	83.08458	86.082474
3.8.2.12	QA training	87.06468	88.60104	85.07463	86.80203
3.8.2.13	QA training	87.31343	86.40777	88.55721	87.469287
3.8.2.14	QA training	88.30846	86.66667	90.54726	88.564477
3.8.2.15	Thread training	76.51515	50	83.87097	62.650602
3.8.3	QA training	88.30846	86.66667	90.54726	88.564477
	Thread training	77.27273	50.98039	83.87097	63.414634
3.8.3.1	QA training	89.05473	88.29268	90.04975	89.162562
3.8.3.2	QA training	89.05473	88.29268	90.04975	89.162562
	Thread training	77.27273	50.98039	83.87097	63.414634
3.8.3.3	QA training	87.31343	86.40777	88.55721	87.469287
3.8.3.4	QA training	89.05473	89.05473	89.05473	89.054726
3.8.3.5	QA training	89.30348	89.10891	89.55224	89.330025
3.8.3.6	QA training	89.30348	88.72549	90.04975	89.382716
3.8.3.7	QA training	89.801	89.21569	90.54726	89.876543
3.8.3.8	QA training	88.80597	89	88.55721	88.778055
3.8.3.9	QA training	91.30435	93.71728	89.05473	91.326531
3.8.3.9	Single training	71.78218	71.875	97.87234	82.88288
3.8.3.9	Thread training	76.51515	50	90.32258	64.36782

Appendix B

Expanded Algorithms

B.1 Text Cleanup Algorithm

```
BEGIN text_cleanup

  sub slang_handler(text):
    for each word in text:
      if word in slang dictionary:
        replace word with slang definition

    return text


  sub process_niv(token):
    token = shorten elongated characters such as hellooooo

    if token is in-vocabulary:
      return token

    suggested_token = PyEnchant suggested word
    return token from suggested_token with best distance score


  sub capital_percentage(tokens):
    capital_percentage = []
    for each token in tokens:
      cap = 0
      if token is alphanumerical and not title case:
        cap = count capital percentage
      push cap to capital_percentage

    return capital_percentage


  sub remove_stopwords(pos_tag, tokens)
    for each word in pos_tag:
      if word in stopwords dictionary:
        remove word from pos tag and tokens

    return pos_tag, tokens


  sub clean_text(text):
    punctuation_count = count punctuation in the text

    filter text from every non-boundary punctuations

    resolve pronoun like im to i'm or ur to you're
```

```
    text = slang_handler(text)

    tokens = tokenize text using nltk tokenize

    capital_percentage = get_capital_percentage(tokens)

    for each token in tokens:
      if token is laughter sound:
        replace token with simple representation of laughter
      else:
        if token is not-in-vocabulary:
          process_niv(token)

    pos_tag = pos tagging token using nltk pos tag function

    pos_tag, tokens = remove_stopwords(pos_tag, tokens)

    return pos_tag, tokens, capital_percentage, punctuation_count

  sub emoticon_handler(text):
    find emoticon in text using emoticon dictionary

    new_text = remove text from emoticon

    emoticon_sentiment = calculate average score of all emoticons found

    return emoticon_sentiment, new_text


  for each r in inputs:

    if input type is single text or thread:
      text, emoji = separate emoji from text

      emoticon_sentiment, text_only = emoticon_handler(text)

      text_details = clean_text(text)

    else if input type is QA:
      text_array = split r using QA delimiter

      for each text in text_array:
        text, emoji = separate emoji from text

        emoticon_sentiment, text_only = emoticon_handler(text)

        text_details = clean_text(text)

      pair text_details of Q and A together

  return whole cleaned up document

END
```

B.2 Text Processing Algorithm

```
BEGIN process_text

  sub apply_modifier(index, score):
    if capitalization_percentage > 0.5:
      if score < 0:
        score = score - scalar
      else if score > 0:
        score = score + scalar

    while true:
      search backward and forward for modifier word until boundary found

      if modifier word found:
        apply modifier to score

    return score


  sub but_check(sentence_score):
    if token "but" found:
      decrease every sentiment score for each word found before "but"
      increase every sentiment score for each word found after "but"

    return new_score


  sub get_sentiscore(text):
    for token in text:
      get sentiment_score from sentiwordnet

      if sentiment_score exists:
        token_sentiment_score = apply_modifier(index, score)

        push token_sentiment_score to sentence_score

    sentence_score = but_check(sentence_score)

    return average sentence_score


  sub check_aggression(token, text):
    if token in aggression_dictionary:
      if token is noun:
        return true
      else:
        search backward and forward for second or third person pronoun
        if pronoun found:
          return true
        else:
          return false
    else:
      return false
```

```
  sub get_aggression_score(text):
    for token in text:
      if check_aggression(token, text) is true:
        token_sentiment_score = -1
        token_sentiment_score = apply_modifier(index, score)

        push token_sentiment_score to sentence_score

    sentence_score = but_check(sentence_score)

    aggression_score = sum of sentence_score

    if any laughter present:
      aggression_score = aggression_score + 1

    return aggression_score


  sub get_positive_score(text):
    for token in text:
      if token in positive_word dictionary:
        token_sentiment_score = 1
        token_sentiment_score = apply_modifier(index, score)

        push token_sentiment_score to sentence_score

    sentence_score = but_check(sentence_score)

    return sum of sentence_score


  sentiscore = get_sentiscore(text)
  aggression_score = get_aggression_score(text)
  positive_score = get_positive_score(text)
  emoji_sentiment = emoji_sentiment_analysis(emoji)

  return scores
END
```

References

Akhtar, M. S., Gupta, D., Ekbal, A., & Bhattacharyya, P. (2017). Feature selection and ensemble construction: A two-step method for aspect based sentiment analysis. *Knowledge-Based Systems, 125*, 116-135. doi:https://doi.org/10.1016/j.knosys.2017.03.020

Albertini, S., Zamberletti, A., & Gallo, I. (2014). Unsupervised feature learning for sentiment classification of short documents. *JLCL, 29*(1), 15.

Aslam, S. (2018). Twitter by the Numbers: Stats, Demographics & Fun Facts. Retrieved from https://www.omnicoreagency.com/twitter-statistics/

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. *Proceedings of LREC, 10*.

Bakeman, R., McArthur, D., Quera, V., & Robinson, B. (1997). *Detecting Sequential Patterns and Determining Their Reliability with Fallible Observers* (Vol. 2).

Bastiaensens, S., Vandebosch, H., Poels, K., Van Cleemput, K., DeSmet, A., & De Bourdeaudhuij, I. (2014). Cyberbullying on social network sites. An experimental study into bystanders' behavioural intentions to help the victim or reinforce the bully. *Computers in Human Behavior, 31*, 259-271. doi:https://doi.org/10.1016/j.chb.2013.10.036

Bauman, S., Toomey, R. B., & Walker, J. L. (2013). Associations among bullying, cyberbullying, and suicide in high school students. *Journal of Adolescence, 36*(2), 341-350. doi:https://doi.org/10.1016/j.adolescence.2012.12.001

Bayzick, J., Kontostathis, A., & Edwards, L. (2018). *Detecting the presence of cyberbullying using computer software.* (Distinguished Honors), Ursinus College.

Bobicev, V., & Sokolova, M. (2017). *Inter-Annotator Agreement in Sentiment Analysis: Machine Learning Perspective*.

Cohen, D. (2017). On any given day, 60 million emojis are used on Facebook; 5 billion on messenger. Retrieved from http://www.adweek.com/digital/facebook-world-emoji-day-stats-the-emoji-movie-stickers/

Dehue, F., Bolman, C., & Völlink, T. (2008). Cyberbullying: Youngsters' experiences and parental perception. *CyberPsychology & Behavior, 11*(2), 217-223. doi:10.1089/cpb.2007.0008

Del Bosque, L. P., & Garza, S. E. (2014). *Aggressive Text Detection for Cyberbullying*, Cham.

Desai, N., & Narvekar, M. (2015). Normalization of Noisy Text Data. *Procedia Computer Science, 45*, 127-132. doi:https://doi.org/10.1016/j.procs.2015.03.104

Devika, M. D., Sunitha, C., & Ganesh, A. (2016). Sentiment analysis: A comparative study on different approaches. *Procedia Computer Science, 87*, 44-49. doi:https://doi.org/10.1016/j.procs.2016.05.124

Ekman, P., & Friesen, W. V. (1971). Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology, 17*(2), 124-129. doi:10.1037/h0030377

Engman, L. (2016). *Automatic Detection of Cyberbullying on Social Media.* (Master's in Computing Science), Umeå University, Sweden.

Facebook company info. (2018). Retrieved from https://newsroom.fb.com/company-info/

Full list of emoji v11.0. (2018). Retrieved from https://unicode.org/emoji/charts/full-emoji-list.html

Golbeck, J., Ashktorab, Z., Banjo, R. O., Berlinger, A., Bhagwan, S., Buntain, C., . . . Wu, D. M. (2017). *A Large Labeled Corpus for Online Harassment Research*. Paper presented at the Proceedings of the 2017 ACM on Web Science Conference, Troy, New York, USA.

Gordeev, D. (2016). Automatic detection of verbal aggression for Russian and American imageboards. *Procedia - Social and Behavioral Sciences, 236*, 71-75. doi:https://doi.org/10.1016/j.sbspro.2016.12.022

Gordeev, D., & Potapova, R. (2016). *Detecting state of aggression in sentences using CNN.* Paper presented at the International Conference on Speech and Computer, Cham.

Hanna, R., Rohm, A., & Crittenden, V. L. (2011). We're all connected: The power of the social media ecosystem. *Business Horizons, 54*(3), 265-273. doi:https://doi.org/10.1016/j.bushor.2011.01.007

Hinduja, S., & Patchin, J. W. (2008). Cyberbullying: An exploratory analysis of factors related to offending and victimization. *Deviant Behavior, 29*(2), 129-156. doi:10.1080/01639620701457816

Hogenboom, A., Bal, D., Frasincar, F., Bal, M., de Jong, F., & Kaymak, U. (2013). *Exploiting emoticons in sentiment analysis*. Paper presented at the Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal.

Hogenboom, A., Bal, D., Frasincar, F., Bal, M., De Jong, F., & Kaymak, U. (2015). Exploiting emoticons in polarity classification of text. *Journal of Web Engineering, 14*(1-2), 22-40.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). *A practical guide to support vector classification* (Vol. 101).

Hutto, C. J., & Gilbert, E. (2015). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*.

Jivani, A. (2011). A Comparative Study of Stemming Algorithms. *International Journal of Computer Technology and Applications, 2*(6), 1930-1938.

Joshi, R. (2016). Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. Retrieved from https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Jurafsky, D., & Martin, J. H. (2016). Part-of-speech tagging. *Speech and Language Processing*.

Jurek, A., Mulvenna, M. D., & Bi, Y. (2015). Improved lexicon-based sentiment analysis for social media analytics. *Security Informatics, 4*(1), 9. doi:10.1186/s13388-015-0024-x

Kaushik, C., & Mishra, A. (2014). A scalable, lexicon based technique for sentiment analysis. *International Journal in Foundations of Computer Science & Technology, 4*. doi:10.5121/ijfcst.2014.4504

Khoo, C. S., & Johnkhan, S. B. (2017). Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*. doi:10.1177/0165551517703514

Kietzmann, J. H., Hermkens, K., McCarthy, I. P., & Silvestre, B. S. (2011). Social media? Get serious! Understanding the functional building blocks of social media. *Business Horizons, 54*(3), 241-251. doi:https://doi.org/10.1016/j.bushor.2011.01.005

Konopík, M., & Pražák, O. e. (2015). *Information sources of word semantics methods.* Paper presented at the Speech and Computer, Cham.

Koo, P. S. (2018). Accuracy, Precision, Recall or F1? Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PLOS ONE, 10*(12), e0144296. doi:10.1371/journal.pone.0144296

Landis, J. R., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics, 33*(1), 159-174. doi:10.2307/2529310

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? *Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*.

Marengo, D., Giannotta, F., & Settanni, M. (2017). Assessing personality using emoji: An exploratory study. *Personality and Individual Differences, 112*, 74-78. doi:https://doi.org/10.1016/j.paid.2017.02.037

Matsumoto, S., Takamura, H., & Okumura, M. (2005). *Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees*, Berlin, Heidelberg.

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR, abs/1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality*. Paper presented at the Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Lake Tahoe, Nevada.

Miller, G. A. (1995). WordNet: a lexical database for English. *Commun. ACM, 38*(11), 39-41. doi:10.1145/219717.219748

Musto, C., Semeraro, G., & Polignano, M. (2014). *A comparison of Lexicon-based approaches for sentiment analysis of microblog posts*.

Nalinipriya, G., & Asswini, M. (2015). A dynamic cognitive system for automatic detection and prevention of cyber-bullying attacks. *ARPN Journal of Engineering and Applied Sciences, 10*(10), 4618-4626.
Nandhini, B. S., & Sheeba, J. I. (2015). Online social network bullying detection using intelligence techniques. *Procedia Computer Science, 45*, 485-492. doi:https://doi.org/10.1016/j.procs.2015.03.085
Nielsen, M. (2018). *Neural Networks and Deep Learning*.
Oleszkiewicz, A., Karwowski, M., Pisanski, K., Sorokowski, P., Sobrado, B., & Sorokowska, A. (2017). Who uses emoticons? Data from 86702 Facebook users. *Personality and Individual Differences, 119*, 289-295. doi:https://doi.org/10.1016/j.paid.2017.07.034
Parris, L., Varjas, K., Meyers, J., & Cutts, H. (2011). High school students' perceptions of coping with cyberbullying. *Youth & Society, 44*(2), 284-306. doi:10.1177/0044118X11398881
Paulo. (2017). How to Scrape Facebook Page Posts and Comments to Excel (with Python). Retrieved from https://nocodewebscraping.com/facebook-scraper/
Pettalia, J. L., Levin, E., & Dickinson, J. (2013). Cyberbullying: Eliciting harm without consequence. *Computers in Human Behavior, 29*(6), 2758-2765. doi:https://doi.org/10.1016/j.chb.2013.07.020
Power, A., Keane, A., Nolan, B., & Neill, B. (2017). A lexical database for public textual cyberbullying detection. *2017, 23*(2), 30.
Ray, S. (2017). 6 easy steps to learn Naive Bayes algorithm (with codes in Python and R). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
Reynolds, K., Kontostathis, A., & Edwards, L. (2011). *Using Machine Learning to Detect Cyberbullying*. Paper presented at the Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops - Volume 02.
Rezvan, M., Shekarpour, S., Balasuriya, L., Thirunarayan, K., Shalin, V., & Sheth, A. (2018). *A quality type-aware annotated corpus and lexicon for harassment research*.
Ruths, D., & Pfeffer, J. (2014). Social media for large studies of behavior. *Science, 346*(6213), 1063-1064. doi:10.1126/science.346.6213.1063
Saif, H., He, Y., Fernandez, M., & Alani, H. (2016). Contextual semantics for sentiment analysis of Twitter. *Information Processing & Management, 52*(1), 5-19. doi:https://doi.org/10.1016/j.ipm.2015.01.005
Schoffstall, C. L., & Cohen, R. (2011). Cyber aggression: The relation between online offenders and offline social competence. *Social Development, 20*(3), 587-604. doi:10.1111/j.1467-9507.2011.00609.x
Shirani-mehr, H. (2015). *Applications of deep learning to sentiment analysis of movie reviews*.
Sim, J., & C Wright, C. (2005). *The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements* (Vol. 85).
Singh, J., Singh, G., & Singh, R. (2017). Optimization of sentiment analysis using machine learning classifiers. *Human-centric Computing and Information Sciences, 7*(1), 32. doi:10.1186/s13673-017-0116-3
Slonje, R., & Smith, P. K. (2008). Cyberbullying: Another main type of bullying? *Scandinavian Journal of Psychology, 49*(2), 147-154. doi:10.1111/j.1467-9450.2007.00611.x
Slonje, R., Smith, P. K., & Frisén, A. (2013). The nature of cyberbullying, and strategies for prevention. *Computers in Human Behavior, 29*(1), 26-32. doi:https://doi.org/10.1016/j.chb.2012.05.024
Smith, P. K., Mahdavi, J., Carvalho, M. M. d., Fisher, S., Russell, S., & Tippett, N. (2008). Cyberbullying: Its nature and impact in secondary school pupils. *Journal of child psychology and psychiatry, and allied disciplines, 49*(4), 376-385.
Stacey, E. (2009). Research into cyberbullying: Student perspectives on cybersafe learning environments. *Informatics in Education, 8*, 115-130.
Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics, 37*(2), 267-307. doi:10.1162/COLI_a_00049
Talwar, V., Gomez-Garibello, C., & Shariff, S. (2014). Adolescents' moral evaluations and ratings of cyberbullying: The effect of veracity and intentionality behind the event. *Computers in Human Behavior, 36*, 122-128. doi:https://doi.org/10.1016/j.chb.2014.03.046

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). *Feature-rich part-of-speech tagging with a cyclic dependency network*. Paper presented at the Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, Edmonton, Canada.

Tul Ain, Q., Ali, M., Riaz, A., Noureen, A., Kamran, M., Hayat, B., & Rehman, A. (2017). Sentiment analysis using deep learning techniques: A review. *International Journal of Advanced Computer Science and Applications (ijacsa), 8*(6), 10.

Whittaker, E., & Kowalski, R. (2014). Cyberbullying via social media. *Journal of School Violence, 14*(1), 11-29. doi:10.1080/15388220.2014.949377

Xu, J.-M., Jun, K.-S., Zhu, X., & Bellmore, A. (2012). *Learning from bullying traces in social media*. Paper presented at the Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Montreal, Canada.

Yin, D., Xue, Z., Hong, L., Davison, B. D., & Edwards, L. (2009). *Detection of harassment on Web 2.0.* Paper presented at the Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009, Madrid, Spain.